advantere
School of
Management

Enhancing Option Pricing Using Machine Learning and Black-Scholes Model

Autor: Emanuele Ospina
Tutor: Dr. Luba Schoenig & Jorge Martin

Capstone Project
Advantere School of Management
Master in finance

Madrid
July 2024

Executive Summary

The thesis "Enhancing Option Pricing Using Machine Learning and Black-Scholes Model" aims to improve option pricing accuracy by integrating machine learning techniques with the traditional Black-Scholes model. The main objective is to address the Black-Scholes model's limitation of assuming constant volatility by incorporating dynamically predicted volatility using machine learning models. Historical financial data, including stock prices, macroeconomic indicators, and technical indicators, were collected to train and validate Random Forest and Neural Network models for predicting market volatility.

The Black-Scholes model was modified to include these dynamically predicted volatilities, and its performance was compared to traditional methods and the GARCH model. The study concluded that the Neural Network model performed better than the Random Forest model, with a lower Root Mean Square Error (RMSE) of 8.63, indicating higher accuracy and better trend-following capabilities. Both models effectively captured market trends, but the Neural Network model was more consistent and responsive to rapid market changes.

Future research should explore hybrid models combining the strengths of different machine learning techniques to enhance predictive performance further. Adding more market indicators and employing rigorous validation methods will improve model accuracy and responsiveness.

Keywords

Option Pricing, Black-Scholes Model, Machine Learning, Financial Modelling, Volatility Prediction, Neural Networks, Random Forests, Market Volatility, Predictive Modelling, Data Integration

Table of Contents

Table of Figures

Table of Equations

1. Introduction

1.1 Project Context and Goals

Pricing options is one of the defining actions in the financial market, which affects trading strategies and risk-taking of a market agent and also one's investment behaviour. First introduced in 1973, the Black-Scholes model has become the most wide-spread theoretical option pricing formula. This model is implemented for the calculation of the price of European options (Black & Scholes, 1973). The applicability of this model is based on the no arbitrage theory and the possibility of investing long-term risk-free is real, even with a riskless underlying and optionary asset owing to the existing market.

$$C = S_0 N(d_1) - K e^{-rT} N(d_2)$$

$$P = K e^{-rT} N(-d_2) - S_0 N(-d_1)$$

$$where \;\; d_1 = \frac{ln\left(S_0/K\right) + \left(r + \sigma^2/2\right)T}{\sigma\sqrt{T}}$$

$$d_2 = \frac{ln\left(S_0/K\right) + \left(r - \sigma^2/2\right)T}{\sigma\sqrt{T}} = \; d1 \; - \sigma\sqrt{T}$$

Equation 1. Black - Scholes Model

However, the model widely applied and accepted within academic circuits, is highly criticized due to the critical limitations and drawbacks based on fundamentally baseless assumptions. The critical drawback of the formula is based on the model's assumptions, such as constant volatility and log-normal distribution of underlying asset prices, which have been questioned for more than four decades. The scope of this work lies in tackling the limitations of the option pricing model by enhancing it with machine learning instruments.

The main aim of the project is to create a method that would increase the accuracy of option pricing through the appliance of machine learning when working with a Black-Scholes model. The rationale behind this choice lies in one of the major issues of this model, the idea of a certain volatility level that remains constant while assessing an option. It represents a strong valid point as a long continuous experience indeed maintains a uniform tendency, but it cannot be used for giving workable foresight in the quickly-shifting environment.

1.2 Scope and Research Question

This project is a part of the larger movement towards the use of advanced computational methods to gain understanding of complex problems within financial economics. More specifically, the set boundaries of this project include the integration of machine learning models with the Black-Scholes formula for the purpose of predicting and employing stochastic volatility for the purpose of pricing European options. The question of this research should be as follows:

"Will the integration of machine learning models for predictive purposes increase the accuracy of the Black-Scholes option pricing formula?"

This will serve as the guiding question through the research and will be resolved in the concluding section of this project.

1.3 Methodology

In order to answer the research question, this project uses a hybrid methodology that bridges traditional financial theory with modern, advanced machine learning. The process begins with a deep review of the literature to form a theoretical base and understand the limitations of the existing approaches. Afterward, it is as follows:

a. Data collection
b. Feature engineering
c. Machine learning model creation
d. Model integration with the Black-Scholes approach.
e. Validation and performance assessment.

1.4 Project Contents Overview

My literature review will firstly cover prior research on the Black-Scholes model and its applications and limitations. The past work on integrating machine learning into financial modelling will also be explored.

In methodology, how data were collected and engineered, and what machine learning models were selected, trained, and integrated into pricing options will be presented.

The results & discussion section will show the results using machine learning and the integrated models, and their accuracy and performance will be discussed.

The conclusions & future work will answer the question of the research projects and the implications and limitations behind that answer. Areas for future research will also be suggested.

This project is beneficial, as it is not only concerned with the abstract implications of option pricing, but also the connection between traditional analytical and modern computational data modelling techniques.

2. Literature Review

2.1 Introduction to Option Pricing Models

The pricing of options is one of the basic parts of financial mathematics, as it offers important methods to price financial derivatives. One of the important days for the first time has occurred in the mid-seventies, when Fischer Black and Myron Scholes published the Black-Scholes model in 1973 (Black & Scholes, 1973). Fischer and Myron were the authors who first made the closed form for the pricing of European options public. Among other things, they postulated that the prices of stocks develop under the rules of geometrical Brownian movement and that there is no opportunity for arbitrage. Enabling their pricing was, therefore, first simpler and, second, feasible in practical terms.

2.2 Theoretical Foundations of the Black-Scholes Model

The Black-Scholes model is based on some assumptions: the logarithmic returns of the underlying asset are distributed normally, markets are no-frictional, the risk-free rate is constant, and the volatility of the underlying asset remains perpetuated through the life of the option (Merton, 1973).

2.3 Limitations of the Black-Scholes Model

The Black-Scholes model has been a critical breakthrough in theoretical finance, especially in the field of pricing European options. Indeed, it has become a cornerstone on which a large part of modern financial engineering is based due to the simplicity of its formulation. Nonetheless, the application has also illustrated several major difficulties in this model. Many of them are connected to the assumptions of its building: these assumptions rarely correspond to the state of a real market, which often results in mispricing and risk mismanagement, two elements so important in trading and hedging processes.

2.3.1 Assumption of Constant Volatility

Nevertheless, even the creators of the model assumed that the Black-Scholes model was a failed model as it based on too many assumptions, which is almost impossible to fulfil have the asset. Unfortunately, the Black-Scholes model is still used in option pricing, and often options are simply inappropriately expensive or cheaply evaluated. It is explained by one reason, the most impossible parameter in the Black-Scholes model is the constant volatility. The model means that volatility of the underlying asset return is constant, known and does not change during the option's life (Heston, 1993).

However, the volatility is not a constant parameter; it is influenced by different factors like market sentiment, economic events, company-specific news and etc. Then, it is logical that there might be an error in the volatility estimate which then would lead to a huge error in the option pricing. Moreover, there can be times of huge financial stress, e.g., 2008, where the volatility can spike dramatically, which the Black-Scholes model cannot predict. Therefore, the option can be strongly under- or overpriced based on the current market situation, and it is clear who will lose, the trader and financial institutions.

2.3.2 Log-Normal Distribution of Stock Prices

Stock prices can only be positive in reality, which means that rates of return follow a log-normal distribution. Although the assumption is convenient, it is inconsistent with the nature of financial return distributions. The real killer of the normal return assumption is that it is perfectly acceptable for financial return distributions to have fatter tails and to exhibit excess kurtosis (Tsay, 2005).

In the real world, financial markets often undergo an event, shocking the system. Events are shock to the stock system which causes distribution to fail as normal. Financial markets always encounter regular shock events, such as economic announcements, geopolitical events, major corporate moves, such as M&A, leading to huge jumps in prices. Events are shocking to the system, and the model here misprices further out-of-the-money options by underestimating the likelihood of these extreme events.

2.3.3 Constant Risk-Free Rate

Secondly, the Black-Scholes model also assumes another unspecified condition: the risk-free rate is known and constant. This statement is often interpreted for an estimated rate based on the government bond yields (Hull, 2017). However, this assumption is barely correct, as it is entirely possible that the risk-free rate may not be described as a constant one. Given the changes in major policy action of the central bank, measures of inflation, and other factors, the interest rate is most certainly volatile.

Since the change of the risk-free rate would influence the present value of the strike price of the option and the future asset price. That value was added into the function, the stability of that value could not be stable. As a result, the adjustment of the value of interest rates eliminates the possibility of fluctuating the share prices based on their real impact, which allows to further make the same options cheaper to buy.

The Black-Scholes model is used with the 3-month (3m) Treasury bill rate as a risk-free rate reference in this project. Though 3m Treasury carries some country credit risk, it is also a generally well-accepted benchmark in financial modelling because its historical data and market liquidity. On the other hand, this could be viewed as a suitable risk-free rate proxy (similar to SOFR) because SOFR is market-set and therefore credit risk-free by being fully collateralized. Nevertheless, the current study keeps 3m Treasury for traditionalism and comparability with another relevant research. This is because of its wide-spread use in both academic research and practice, providing a consistent and fairly standardized criterion for calculating risk-free rates.

2.3.4 No Dividends

It is important to understand that the Black-Scholes formula does not consider the dividend yield of the stock. In other words, if the stock is expected to pay dividends, the expected value of the stock, therefore, decreases (Black & Scholes, 1973). In addition, the value of the dividend yield also matters for a given option – in the moneyness and time value calculations. These indicators are also not an exception for high dividend securities for derivative calculations.

2.3.5 Market Frictions

The Black-Scholes model does not take into account transaction costs or taxes and implies a continuous nature of trading (Black & Scholes, 1973). Indeed, transaction costs, taxes, and the discrete nature of trading are not immaterial for Black-Scholes model trading strategies, which enables to overestimate its profitability and underestimate/ignore a true risk.

2.4 Advancements Beyond Black-Scholes

Several models were developed to avoid these flaws altogether by providing more possibilities of the option pricing process. The most used model is stochastic volatility model, which also perceives the volatility to be a stochastic process (Heston, 1993). It means that the volatility is capable of changing and primarily depends on the market conditions. This, however, is still the flawed application, as the general market behaviour showed that models in which it is based, such as Heston model, present the more realistic market behaviour and therefore may lead to the more applicable market volatility estimations.

However, the variation of this model is also possible to use and some alternative fits are available, such as jump-diffusion models, where the jump integrates in the price dynamics. This application shows the sudden event occurring in the market that leads to the extreme change in an asset price distribution. It means that there is a jump up or down, and it usually coincides with the extremely good or bad news. It proves that heavy market impact or the significant corporate events is the frequent time of jump. Thus, the model provides the possibility to adjust to the actual asset returns distribution and allows pricing the options during heavy chaos periods.

Local volatility models consider the volatility to be dependent on the time and price levels. The first model was introduced by Dupire and shows that the model can be parameterized according to the time and price the volatility for the option periods considered to be impossible to imagine with Black-Scholes model.

2.5 Integration of Machine Learning in Financial Modelling

Actually, the existence of machine learning tools and their development in the sphere of financial modelling significantly contributed to more adaptive and reliable while analytically faster ones. Machine learning offers very powerful and efficient tools that can deal with the massive amount of data, elaborate complicated patterns and create insightful predictions escaping the limits of traditional ones or being beyond their practical capabilities at all. This is illustrated by the fact that machine learning tools have been employed to address traditional financial modelling problems such as the exhaustion of the potential of Black-Scholes model.

2.5.1 Overview of Machine Learning in Finance

Finance is a vast and diverse industry in which machine learning is used not only in algorithmic trading and risk management but also in asset management and pricing options, and many other sub-behaviours (Lopez de Prado, 2018). Machine learning models can process big and fast data and learn by example rather than by instructing how to write code, which makes them a natural-best fit for financial markets that tend to be very dynamic and sometimes insanely chaotic (Goodfellow, Bengio & Courville, 2016). Machine learning works well with options since it can model complicated non-linear relationships and be sensitive to new data.

2.5.2 Enhancing Volatility Forecasting

The first limitation is that the Black-Scholes model works with constant volatility. In this sense, it is possible to apply machine learning models, including those for time-series forecasting. For example, Generalized Autoregressive Conditional Heteroskedasticity uses historical market data to forecast volatility. However, there are several reasons why machine learning is better.

First of all, Long Short-Term Memory Networks are the architecture of the Recurrent Neural Networks with the enhancement. It is easy to see that the more extended period, the wider the spectrum. Of course, moreover, the volatility of periods may be different in strength of actions from one patter to another. In this regard, LSTMs are valuable because they predict these intervals better.

Secondly, Random Forests and Gradient Boosting Machines are ensemble learning techniques suitable for classification and regression problems. Since market data is non-dependent linearly or even independent, scrolling through decision trees in this regard may not always be the most effective way to represent volatility.

2.5.3 Predictive Modelling for Option Pricing

Finally, machine learning can also contribute to the problem indirectly by predicting price movements of the underlying or the option itself. Specifically, modern machine learning models can find patterns in historical price movement data and other relevant financial indicators that would be impossible to discover using traditional methods. In this case, the important elements of such a model would be:

Features feared engineering – traditional values in this area include several technical indicators such as moving averages, RSI or relative strength index, MACD or meaning average convergence divergence, fundamental factors such as the earnings ratios as well as market sentiment indicators based on final news sentiment or social media activity.

Neural networks – in addition to Lerner short-term memory, suggested many other types of neural networks trained on historical price and other relevant data in this system to predict future prices of an option. These types include convolution neural networks (CNN) and feed forward neural networks. Neural network can process millions of dimensions of input, understand non-linear interactions between them and ultimately protect complex outputs such as option price for different strike pieces and maturities.

2.5.4 Parameter Optimization and Model Calibration

Machine learning methods optimize Black-Scholes model parameters such as volatility, or calibrating the whole model to better adapt to market data. For this purpose, genetic algorithms, deep reinforcement learning, and Bayesian optimization are used to dynamic parameters adjustment based on new data. Genetic algorithm can optimize using historical option trades given dataset to select optimal set of model parameters that reduce option price errors.

In addition, deep reinforcement learning trains models to make squared decisions. It strives to learn optimal actions in the simulated trading environment, to maximize rewards, that is, to minimize pricing errors or to maximize trading profit.

2.5.5 Challenges and Opportunities

Unfortunately, in addition to the high potential of machine learning in the field of financial modelling, there is a high chance that it will fail. These are overfitting, that it is when the model as a result of training in the answer has memorizing metrics significant drawbacks. “Overfitting is the result of training a model on much more data than the model is capable to…”, a large amount of calculation by almost all ML algorithms and the definition of “black box” models is one of the “news” ones; it is basically the essence of ML – these models are impossible to interpret for anyone, even scientists who have developed and trained them. The ways to solve these problems may be long validation, selective choice of model and regular updating of the same model.

3. Model Development

3.1 Data Collection

In this study, the S&P 500 index is taken as the underlying asset for the options pricing model. The S&P 500 index is chosen for several reasons. Firstly, the S&P 500 index is highly liquid, and information is revealed stock-wise (Hull, 2017). Therefore, a reliable dataset with less noise and fewer outliers can be obtained. Secondly, the S&P 500 is representative of a broad market exposure and a standard benchmark in the finance field. Information includes the historical stock market close prices, high prices, low prices, opening prices, and trading volumes. The data is taken from direct websites or major financial market information providers, such as Bloomberg, Yahoo Finance.

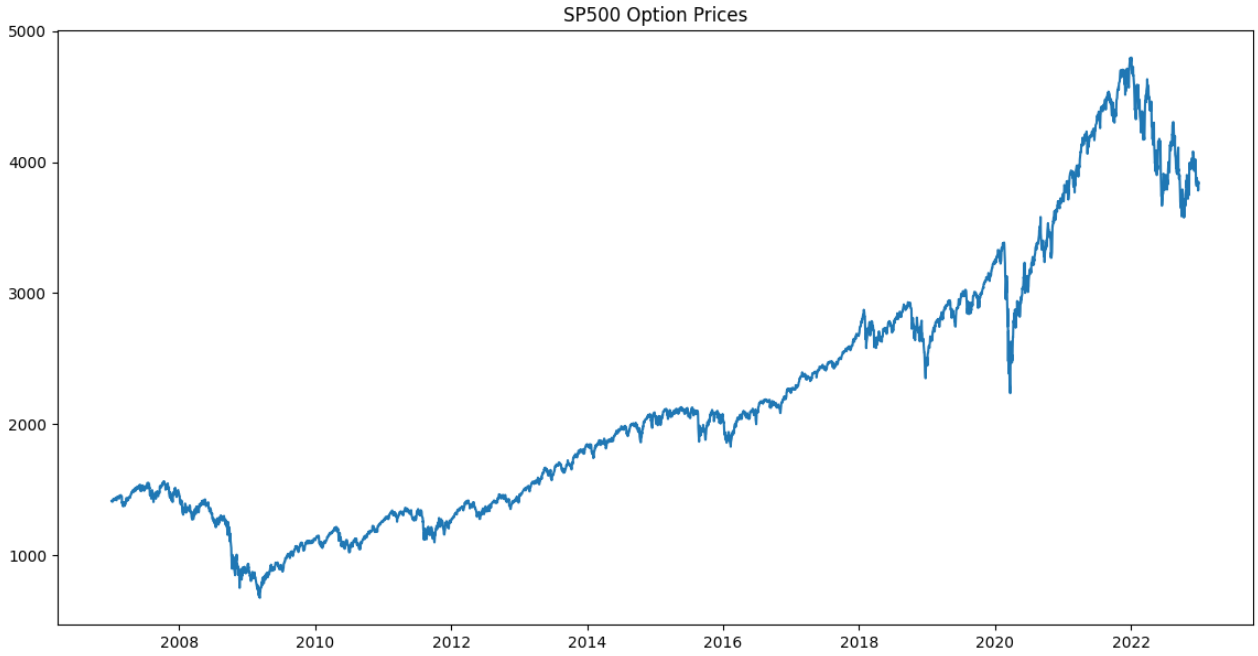

Figure 1. SP500 Option Prices

Further, the macroeconomic indication data, such as interest rates provided by the Federal Reserve and GDP growth rates from the governmental economic reports.

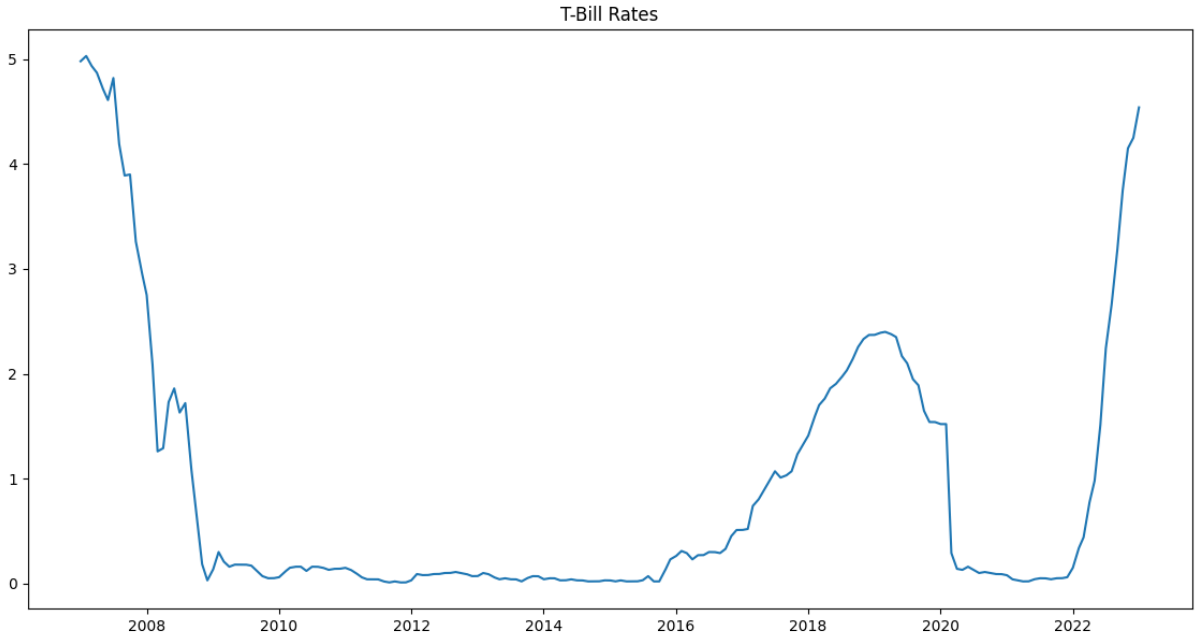

Figure 2. T-Bill Rates

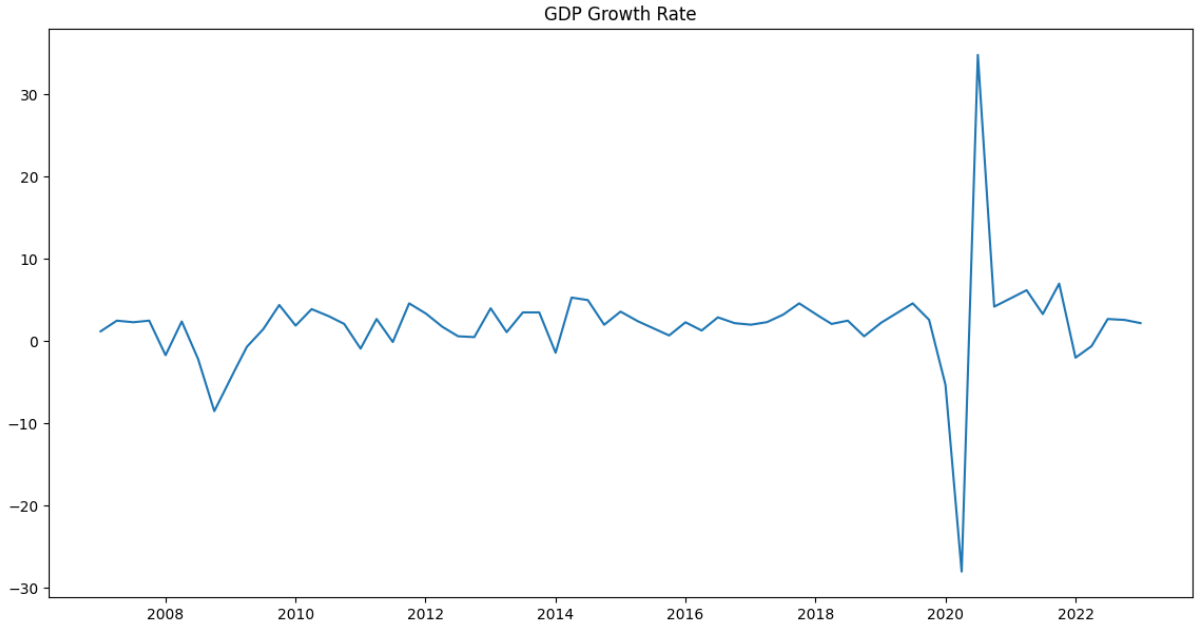

Figure 3. GDP Growth Rate

Moreover, VVIX is going to be used to evaluate the general market sentiment to the S&P 500.

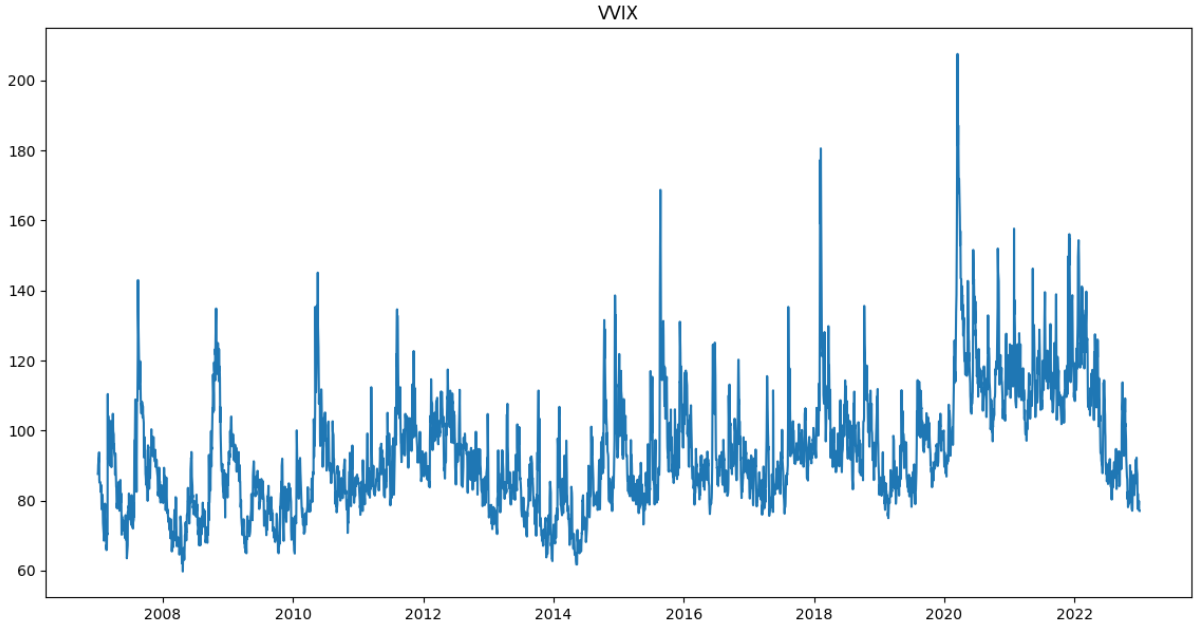

Figure 4. VVIX

The data time is the past fifteen years, which may allow me to obtain different market conditions, including bull or bear market, and assist in developing a model that can withstand different economic cycles.

3.2 Feature Engineering

Before inputting the raw data into any machine learning model, we should pre-process the raw data. The stock price and volume are normalized using both a min-max scaling technique achieved various common scales and maintains the actual difference from and around the value variables; Moreover, we fill in the missing data point as noted; we further maintain the last observation, carried forward for short gaps of the other points for longer periods.

We also engineer additional features, for the model to enhance the prediction power known as the RSI and exponentially weighted moving averages (Murphy, 1999); these include such technical indicators such as the moving averages, exponential moving averages; RSI, for example, capture down or upstream and market price reversal; another include shifted implied volatility trends and another correlated economic indicator such as the change in the interest rates features engineering were used as feature wells. Using exploratory data to correlates with a feature selection automation process, involving well-check mode multicollinearity and well model linear models. Properline models features identified the

multicollinearity in some of the features we already and remain with our existing predictive power and exclude the unnecessary features for the first assault.

3.3 Machine Learning Model Selection

The possibility of the methods selected in the previous section is determined due to the nature of financial markets that can be described as complex and non-linear (Bishop, 2006). Thus, the model that can capture sequence models is Long Short-Term Memory Neural Networks LSTMs. These models can describe time series of the stock price movements, while decision trees of their ensemble versions, for instance, Random Forests can relate part of the non-linear relationship and be robust to overfitting.

These models will be utilized with economic indicators and the market sentiment data. The support vector machines SVMs are able to work in high-dimensional spaces, which is useful for financial data, augmented with multiple presumably informative variables. All these models will be further fine-tuned on a subset of the data to optimize the model number of layers and the structure and depth of trees. Such fine-tuning is crucial for the further optimization of the learning process.

3.4 Model Training and Validation

The model training is conducted on the dataset division into the training 70%, validation 15%, and test 15% segmentation to allow the model to maximize training on most of the data to be able to see and evaluate the models only on the unseen data (Hastie, Tibshirani & Friedman, 2009). The latter is more representative of what you realistically have as you want the model to perform well in the future, namely unknown market conditions. To enable and allow the model to train and use as much available data for training as possible while strictly validating it, K-fold cross-validation is used during the model's training phase.

This strategy also helps to prevent over-fitting, which is the key common problem in machine learning model training due to too complex neural networks. As to the performance metrics suitable for evaluation, the choice of metrics depends on what performance you want to value during the formation of each of the models within the project. For example, the

accuracy of model predictions of price increases is assessed using Mean Squared Error. Model accuracy in determining the direction of price movement is assessed like using accuracy.

Furthermore, when it comes to finance, probability often lies under model outputs, and AUC-ROC curve is the average mix of sensitivity and one-specificity for all possible probability thresholds and can give an insight into valuing the model regardless of its probability output specifics.

4. Integration with Black-Scholes Model

4.1 Modification of the Black-Scholes Model

The proposed change of the Black-Scholes model makes a revolutionary change in the model's initial structure. The model's framework is completely transformed, as opposed to the traditional approach (Heston, 1993). Incorporation of the dynamically changing volatility, which is predicted by a machine learning model, into the Black-Scholes formula makes the model continuously reflect the current market situation, providing a more realistic way of options pricing.

Python-based machine learning models run on a computer and predict future volatility based on a number of parameters, including historical S&P 500 prices and trading volumes, economic indicators, and market news sentiment analysis. It is a central part of the given methodology since the Black-Scholes formula's modification is impossible without machine learning models that predict implied volatility dynamism.

4.2 Implementation Details Using Python

Further examine the implementation in Python, which will be more detailed and thorough, considering the step-by-step process, utilized Python libraries, and how the Black-Scholes model and machine learning relate to each other in a detailed manner (Lopez de Prado, 2018).

4.2.1 Data Handling and Pre-processing

Since the beginning, Python's Pandas library is used for the data acquisition and pre-processing stage since it is a high-performance data manipulation support and strongly competes with others. Fortunately, the high performance does help because, with large datasets, more computations must be made, which with the Panda's help is made simpler. Especially in the financial field, it is very good because it has native support of "time" for indexing and, as is known, the financial data is almost in every case typed into a time series.

Data Cleaning – Missing values are filled and anomalies are corrected. Useless data points are always filtered out. This is done either with the.fillna() function to fill data or with.dropna() to remove incomplete Time Intervals.

Feature construction – For us, in this case, regulators use the future raw features to “predict” the next days. In addition, additional information not present in the raw data is needed to make a “perfect” daily investment decision. That information is provided by technical indicators. All qualifying and disqualifying indicators are included in the list of indicators In this case, the moving averages in particular help to follow trends or follow the S&P 500 volatility patterns.

4.2.2 Machine Learning Model Development

Scikit-Learn is the most frequently used in creating machine learning models because the tools are so wide. At the same time, tools are available for quick and efficient building of models to schedule model learning to the greatest possible extent while continuing effort and time to check if further models can really be training; alternatively. From a straightforward metric and vector like linear regression to the more complicated like ensembles or neural formulas possible several more models. sklearn.tree is suggested for the decision tree and sklearn.ensemble for random forests since they are intended to be accurate and accessible, easy to use tools that have flexibility while building a model and later analysing it. For learning complex patterns and dependencies, Keras with a TensorFlow backend is particularly considered, specifically predicting dynamic volatility in the study. Since the design patterns of LSTM networks can correctly convey sequence-dependent designs, it seems to be a suitable. In addition, stack layers and tunning of hyperparameter using Keras API are performed. TensorFlow backend is used to perform computations since it is GPU accelerated.

4.2.3 Integration with Black-Scholes Model

After training several machine learning models to predict volatility, the resulting predictions will be transferred to the Black-Scholes formula modified in Python. To make it possible, a function in Python that behaves as a Black-Scholes formula wrapper will be developed: it accepts dynamic volatility as an argument, along with other required variables: stock price, strike price, fiat value token price, and time to expire. Numpy is chosen for its excellent mathematical features, which allow me to optimize the valuation formula computation speed in order to be able to compute Black-Scholes relatively quickly. Numpy is excellent for

vectorized calculations, which is rather crucial when pricing options use a relatively large data sample.

4.3 Computational Challenges and Solutions

4.3.1 Enhanced Data Management and Processing

The volumes of the financial information, especially high-frequency trading data or extensive historical information, require sophisticated data management approaches. Although Python's Pandas allows managing the above data, Python's Pandas cannot achieve the same results for real-time analytics:

- Asynchronous data handling: Python apps utilizing asyncio can undertake various tasks at once to improve resource utilization by letting the system undertake other functions while waiting for data to be loaded or processed.
- Multiprocessing and parallel processing: Python's multiprocessing can be used to distribute the data processing to several CBDs. This would be essential to increase the performance of financial indicator computations as well as the performance of machine learning models. For more complex parallel processing and concurrent computing that involves massive datasets or more advanced model training automated routines, libraries such as Joblib and Dask can be utilized.

4.3.2 Optimizing Machine Learning Models with GPU Acceleration

From our user perspective, training complex neural network models LSTMs for volatility prediction need much computation. The following methods are used to offset it:

- Laptop cannot contain 'high-configurations,' but they can have a GPU installed. The use of the NVIDIA CUDA through TensorFlow or PyTorch means model training for deep learning can be faster because the frameworks are engineered to entirely take advantage of GPU hardware acceleration.
- Implement efficient batch processing of data during model training. The GPU memory is optimized in such a manner that it only takes use of GPU memory to an accomplished extent and can use the strongest feasible GPU. Then, managing the batch size is crucial to minor memory overflow while still guaranteeing that the model training is both faster and precise.

4.3.3 Real-Time Data Integration and Model Updating

Therefore, the addition of real-time data in predictive models and recalculation of Black-Scholes in real-time, then the following hard technical features:

- Real-Time Data Pipelines: The first job is to construct a solid pipeline that can store, handle, and transmit the facility with real-time data to value latency. In this project, the following Python's libraries were used in the pipeline, including Apache Kafka, a streaming data platform, and Celery, an asynchronous task queuing platform.
- Dynamic Model Updating: The Black-Scholes model's parameters are altered anytime a new forecast of volatility is viable; it then refactors the most recently available Bootstrap volatility into the Black-Scholes alternative pricing value. This must be completed without occurring downtime and affecting the trading cycle since it could change the nature of the pricing modality.

4.3.4 Mitigating Overfitting and Ensuring Model Generalization

The problem of overfitting is not eliminated since the models have to be trained on historical financial data which is obviously not identical to the future. Nevertheless, the following can be done to address the issue: the training process involves regularization techniques such as L1 and L2, it means that the models also get penalized for being too complex, reducing the overfitting probability for them. Through k-folder cross-validation, the model is tested on different subsets of data to make sure it generalizes and does not merely memorize and perform well on the training set.

4.3.5 Back testing and Continuous Validation

To ensure that the machine learning-enhanced Black-Scholes model performs reliably under various market conditions:

- Rigorous back testing of the historical market data. This has to be accomplished by Python financial and statistical libraries that offer an API, simulation model's API tells how the model would have done over the historical period we have data for.
- Rolling window validation: The model continuously gets re-evaluated makes use of a "rolling window" considering the most recent available market data. The model gets

updated regularly on a rolling basis so that enormous adjustments are not made at one shot.

These systematic installation policies and highly disciplined examination methodologies will make the project successfully integrates the volatility prediction of machine learning in the traditional Black-Scholes model. It not only improves conventional options pricing methods but also improves the model's flexibility and versatility to operate in the modern era marketed.

5. Analytical Framework and Data Integration

5.1 Data Sources and Pre-processing

To apply this methodology our first step was to gather datasets needed for our analysis and pre-process them. The historical closing prices of the S&P 500 index were obtained through the Yahoo Finance website with the help of the “yfinance library”, the data was from January 1, 2007, to January 1, 2023. Such a long time period was selected to cover different market conditions (including the financial crisis of 2008, growth recovery, and more recent market retractions), providing a comprehensive dataset that summarizes a variety of market dynamics.

Also, the VIX and VVIX were linked from Yahoo Finance for the same dates that we obtained the option prices. The VIX is often also known as the markets "fear gauge", as it represents investors’ expectations for near term volatility, however, the VVIX measure the volatility of the VIX, which give us an indication of how stable these expectations are. Market volatility and investor sentiment can be interpreted via these indices and hence are important part of our analysis.

Accompanying the market data, several macroeconomic indicators were introduced: GDP growth rates, unemployment rates, inflation rates, 3-month Treasury-bill rates. Those indicators were provided by trustworthy databases and imported by CSV files. All three indicators offer insight into the wider economic climate that influences the market behaviour. That is why digging a little deeper into why these are really needed for volatility calculations was important to me:

A strong GDP growth indicates a healthy economy which then translates to more investor confidence and less market volatility. On the other hand, slower or shrinking GDP growth would indicate economic difficulties and consequently add uncertainty and market volatility. Adding in GDP growth rates takes the broader business cycle dynamics that influence market behaviour and volatility. Investors might be buying up shares at a discount, while companies use the funds to weather a period of economic distress (for example, countries at war or after it), to stimulate stock markets, and reduce the risk of selling drop shares too much.

Unemployment is a good indicator of the state of the job market. High unemployment may signal economic trouble, and the decreased consumer spending and corporate profits that it implies can lead to increased volatility in the markets. Conversely, when unemployment is low, the economy might be doing well, which could help support markets. Comprehend how this can be the game changer in capturing the economic sentiment and in knowing how much it can change the stability of the market. For instance, as the financial crisis began in 2008 and unemployment rates started to rise, the markets also became more volatile as investor confidence dropped and uncertainty around the economy began to occur.

Inflation rates are a measure of the rate at which the general price level of goods and services is increasing. Except for moderate inflation, which is often taken as a sign of economic growth and rise in prices, high inflation can lead to weakening purchasing power and corporate profit. This, in turn, creates price instability in the market. Deflation, instead, can suggest that the economy is at a standstill. Inflation rates provide us with a valid way to track and maintain the price stability in the economy and this price stability impacts how investors will behave and how the markets would be volatile. Additionally, Inflation reaching high levels could force central banks to fight back against it and raise interest rates, introducing an entire new dimension of risk and potential volatility in the markets.

The 3-month average T-bill rate is often used as a proxy for the risk-free rate in financial models. It represents the yield on government debt securities, and it is affected is by the policies of central banks. T-bill rate changes can move the needle on the cost of borrowing and have a major influence on investment decisions and ultimately on market volatility. For example, if T-bill rates are going up, this might mean that interest rates are generally headed even higher… and that a tighter monetary policy could also tighten market liquidity even more, raising market volatility. On the other hand, if T-bill rates fall then that suggests that the Fed is perhaps taking a tighter stance, which can help stocks but might also give a hint of some economic woes that could bring volatility.

These different macroeconomic indicators offer unique viewpoints on the economic climate. So, by incorporating them into our analysis we get a comprehensive look at all the different factors driving market volatility. During periods of economic uncertainty or recession, GDP

growth may slow, unemployment may increase, inflation may become wildly unpredictable, or T-bill rates could be adjusted to stimulate or stabilize the economy. These adjustments can result in greater market volatility as investors re-evaluate their positions and realign their expectations.

The data that we harnessed originated from several sources and was published with different frequencies, which made it imperative to synchronize it for a common sanity. We reindexed all data series to a daily frequency, which meant we could combine the two datasets seamlessly. This is an important step in time alignment between the datasets we are using for meaningful and correct analysis.

Cubic interpolation was used to handle missing values in these datasets, which in turn reflects the value of the data points by a cubic polynomial (Spline functions). We took this approach because it lets us generate clean, continuous datasets, which helps avoid biases or inaccuracies due to missing data points. Having a dataset that was already filled in, fully allowed us to carry on our analysis with the confidence being that out dataset integrity was in-tacked.

5.2 Technical Indicators

To beef up our dataset, we calculated a multitude of technical indicators, which are widely deployed in finance as tools to forecast future price direction. Mainly, these were the moving averages (MA20, MA50, MA200), the Relative Strength Index (RSI), and the diverse Moving Average Convergence Divergence (MACD) (Murphy, 1999).

Moving averages work by smoothing out price data across defined periods to allow you to identify trends. 20-day, 50-day and 200-day moving averages to depict the short term, midterm, and long-term trends respectively the RSI Determining Factors. RSI is used to predict the speed and the change in terms of the price movements, which can be overbought or oversold, signalling potential price reversion. The MACD compares short-term and long-term moving averages to give an idea about the strength and direction of a trend, respectively, and presents it in the form of a signal line and a histogram.

The use of these technical indicators served the purpose of approaching a wider arrangement of features. With these features, the predictive power of the models was intended to be boosted.

5.3 Log Returns and GARCH Model

In order to evaluate the change of stock price every day, we calculated the log returns of S&P 500 index. Log returns are preferred in financial analysis since they are time additive and yield a dynamically consistent measure of return that includes the compounding factor (Tsay, 2005). Modelling and analysis, in particular, is something they are used for with great success due to this trait.

After that, we used a GARCH (Generalized Autoregressive Conditional Heteroskedasticity) model on scaled log returns. The GARCH (1,1) model is a common tool for modelling conditional volatility in financial time series due to its ability to capture volatility clustering, a feature of financial time series where we observe 'bursts' of high volatility when periods of high volatility are followed by high volatility and periods of low volatility are followed by low volatility. With the GARCH model, we have a benchmark estimate of the conditional volatility, which we use as our reference to compare the predicted volatilities from the machine learning approaches.

We chose a GARCH (1,1) model because it is an approach commonly used and has been shown to be an efficient way to model financial time series volatility dynamics. Might you argue that comparing our machine learning predictions to the estimates from the GARCH model would allow us to rigorously assess whether, compared to the existing yet fairly limited model, our approach actually adds any value or not.

6. Random Forest Implementation

6.1 Feature Selection

In preparing for the Random Forest implementation, we carefully selected features that would provide meaningful inputs to our model. The features included technical indicators (MA20, MA50, MA200, RSI, MACD, MACDSignal, MACDHist) and macroeconomic variables (GDP Growth Rate, Unemployment Rate, Inflation Rate, 3-Month T-Bill Rate, VVIX). These features were chosen based on their relevance and potential impact on market volatility.

Technical indicators were selected for their ability to capture different dimensions of market behaviour, such as trends, momentum, and volatility. Macroeconomic variables were included to account for broader economic influences on market dynamics. By combining these diverse features, we aimed to create a robust model capable of accurately predicting market volatility.

6.2 Train-Test Split

We make a training test split to evaluate how well our model does. The model is trained here by using 80% of data from a dataset and tested using the remaining 20% of the dataset. This split ensures that the model is trained on a large enough dataset and tested on new data to see how well it will do in the real world.

This technique is used in machine learning to avoid overfitting (memorization of the dataset) (that the model simply repeats the trained data) and that the performance of the model on the training data will be the same as the generalization on new unobserved data (not used in training), it can be in easy other words, the model does not "know by heart".

6.3 Model Training

For this volatility prediction task, we selected both the Random Forest Regressor and then later the Neural Networks as the first model The model created was a Random Forest Regressor which is an ensemble learning method for classification, regression, and other tasks that operates by constructing a multitude of decision trees at training time and outputs the class and takes the average of the individual trees across classification or regression outputs. The random forest model, due to its robustness and adaptability to high-dimensional data, is

a very good choice for our case. We controlled reproducibility and stability of the model training by setting the number of trees was set to 1000 and using a fixed random state.

Random Forest algorithm was chosen because of the clear nonlinear and complex interactions. Given that financial markets have a myriad of interacting and dependent factors and relationships, simple methods are unlikely to be able to correctly model the entire system. Given how a Random Forest will use subsets of the data to learn multiple decision boundaries, it works quite well to deal with these complexities.

An interesting characteristic of the Random Forest model is its ensemble behaviour since the final prediction is a result of the combination of the predictions made over all one trees. Then average the predictions of the trees to reduce the variance and hence run less risk of overfitting. Overfitting happens when a model learns the noise in the training data instead of the underlying trend, and models poorly beyond the sample data. By fitting each tree on a slightly different subset of the data, Random forests ensure the trees are as different from one another as possible, reducing the risk of overfitting.

Most importantly, Random Forests allow us to look at feature importance, so we can see which input variables matter the most in predicting volatility. This is especially precious for financial modelling where understanding pivotal for better forecasting and making investment decisions and risk management strategies.

We decided to use 1000 trees that was a compromise between computational efficiency and model performance. In general, the more trees in the forest leads to more stable and accurate predictions because the averaging effect is stronger. But it also increases the cost of computation, thus the selection of an optimal number of trees is important.

By fixing a random state, the results are deterministic which serves as a key feature of Machine Learning research and model validation. Reproducibility ensures that the model works with the same level of quality when it is reused on other data.

6.4 Volatility Prediction

After we have trained our model, we can use this model and predict the volatility on the test set. The estimated volatility values were compared with the predicted volatility values from the GARCH model. This contrast help us determine just how accurate our Random Forest model has been in forecasting market volatility.

This was something that, by comparing the predictions of the model against the GARCH estimates, we could measure the gained increase of our machine learning style. This was important to confirm how well the Random Forest model performed.

6.5 Results Visualization

In the code, we plot the predicted volatility vs the estimated volatility from GARCH model. Scatter plots and line plots were used to make the visual comparison between actual and predicted values clear. It is important to have these plots in order to gain some insight on how the model is able to capture the underlying structure of the data.

Further, we smoothed the predictions by applying cubic spline interpolation to the predicted volatility, to fit a continuous volatility curve. This interpolation is useful in visualizing the predicted volatility over time and comparing the fit with the actual volatility from the GARCH model.

Detail visualization comparison provides us with a framework to see where our models are excelling as well as where additional fine-tuning should occur.

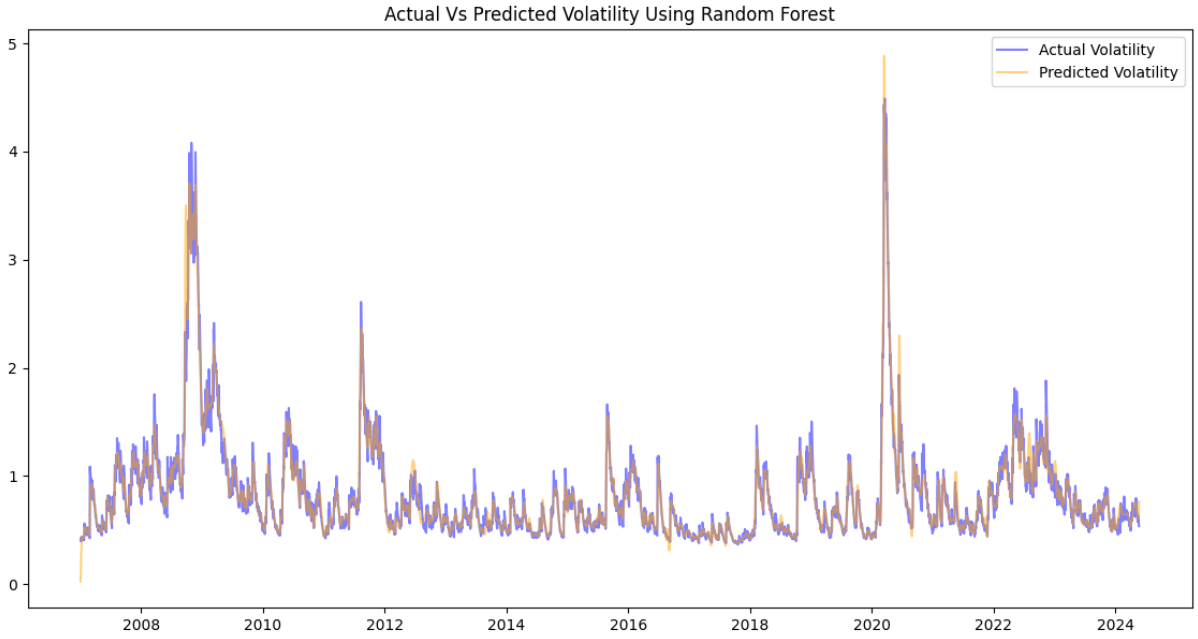

Figure 5. Actual Vs Predicted Volatility - Random Forest

In the image we can see a time series plot that compares the actual market volatility to the volatility predicted by a Random Forest model from 2007 to 2022.

This plot indicate that the Random Forest model tracks the volatility trend quite well with some exception as the predicted volatility pretty much follows the realized volatility. The fact

that this comes out ahead shows the value of a very accurate and reliable model predicting market volatility over a longer period of time.

During the financial crisis of 2008, there is a significant spike in volatility, which is well captured by the model, indicating its ability to adapt to market stress. Similarly, the volatility surge around 2020, corresponding to the COVID-19 pandemic, shows that the model can predict heightened market instability accurately. However, there are periods, such as around 2012 and mid-2019, where the model's predictions slightly overestimate the volatility. This overestimation may be attributed to the model's sensitivity to smaller market fluctuations or external economic events not fully captured by the input features. Otherwise, some underestimations in 2016 suggest that the model might not fully account for unexpected geopolitical or macroeconomic developments during relatively stable periods.

6.6 Black-Scholes Model for Option Pricing

We priced options using the Black-Scholes model with the predicted volatilities from our Random Forest model. Black-Scholes: A common mathematical model used for pricing European options given inputs such as stock price, strike price, time to expiration, risk-free rate, and volatility. Using the predicted volatilities, we estimated the option prices of call options with varying strike prices (97%, 100%, 103% of the current stock price).

This step showed the application of our predicted volatilities in financial markets. That way, we showed how machine learning can be exploited to improve classic financial models by combining the prediction of the Random Forest model with the Black-Scholes formula rewriting the option price based on more accurate and dynamic inputs.

7. Neural Network Implementation

This project uses neural networks because they are very good in capturing complex, non-linear relationships in high dimensional financial data, and also very important when you want to be able to predict the volatility correctly. Work with massive datasets: As the they feature deep architectures and because of their non-linear activation functions, neural networks can work with millions of data points so more complex patterns can be taken into account and better predictive performances can be obtained as compared to more traditional models such as random forests. On the other hand, techniques such as dropout regularization in neural networks can be used to avoid overfitting and improve their generalization ability.

Neural networks come with such benefits over random forests as better scaling with data complexity and size, and superior learning capability via backpropagation and gradient descent optimization. Due to these benefits, neural networks are particularly well suited for the dynamic and noisy appearance of financial data, with the goal of achieving more accurate predictions and more robust forecasts in this work.

7.1 Data Normalization

This procedure guarantees that all features remain with a mean of zero and a standard deviation of one throughout the dataset. Data normalization is so important because it speeds up the learning process of algorithms and it can even improve the performance of many ML models especially neural networks. The StandardScaler from sklearn.preprocessing is used in the code. It uses these all into consideration for that preprocessing. Training data is normalized by fit_transform where mean and std are calculated and then data is scaled. The same scaling parameters are later applied to the test data using the transform function. This guarantees the consistency of images between the training and testing datasets so that the model is able to yield reliable predictions.

7.2 Neural Network Model Creation

A neural network model has to be created where the exact architecture of the model has to be defined like number of layers, how many neurons should be there in each layer and which activation functions should be used. The code defines a feedforward neural network using

the Sequential model in TensorFlow's Keras API. Linear stack of layers, where layers are added in order The first layer is a fully connected layer with 128 neurons and adopts the Rectified Linear Unit (ReLU) activation function. ReLU is often chosen for hidden layers because of the vanishing gradient problem, this can of course helps the training of deep networks. The dropout layers are added to avoid the overfit by randomly setting at the fraction of input units on each update during training to 0. The final layer is an output layer with one neuron and a linear activation function, which is perfect for regression tasks, i.e., the output is a continuous value.

7.3 Model Compilation and Training

After defining the model architecture, we then compile the model and train it. We compile the model which is done using a loss function and an optimizer. Here loss function is nothing but mean squared error, a common loss for regression problems, which computes the average of the square of the difference of the predictions from the ground truth. Adam is the optimizer that was used, which is well known for its efficiency and adaptive representations of learning rate. We train the model using our train data, by running the fit method. Important parameters as being the number of epochs, the batch size or the validation split. A batch size of 32 is used with 20% of the training data set aside for validation to check the model's performance on unseen data while it is training, and the model is allowed to train for 50 epochs.

7.4 Making Predictions and Evaluating the Model

After the model has been trained, the model has to be tested on the test data. To make a prediction, you call the predict method or you evaluate the model based on selected evaluation metrics such as MSE (mean squared error), R-squared score (R^2). Where, MSE calculates the average of the squares of the errors i.e., the average of the squared differences between the actual and the predicted values. Smaller the MSE value lower is the better model performance. R^2 score, or the coefficient of determination, basically tells us what is the % of variance that we can predict the dependent variable knowing the independent variables. An R^2 score of 1 indicates perfect predictions, whereas closer to 0 is poor predictions.

7.5 Plotting Loss Values

We can visualize the training to track if the model is overfitting/underfitting or the learning behaviour of the model as it is being trained. A plot of the value of the loss on the training dataset and the validation dataset against the number of epochs is useful to better understand the learning curve of the model. Both training and validation loss should ideally decrease and converge together as shown below. If you notice your validation loss rising while your training loss continues to fall, then it is an overfitting indicator. Matplotlib is used to plot Loss values in the following code for seeing how your model is performing during Training.

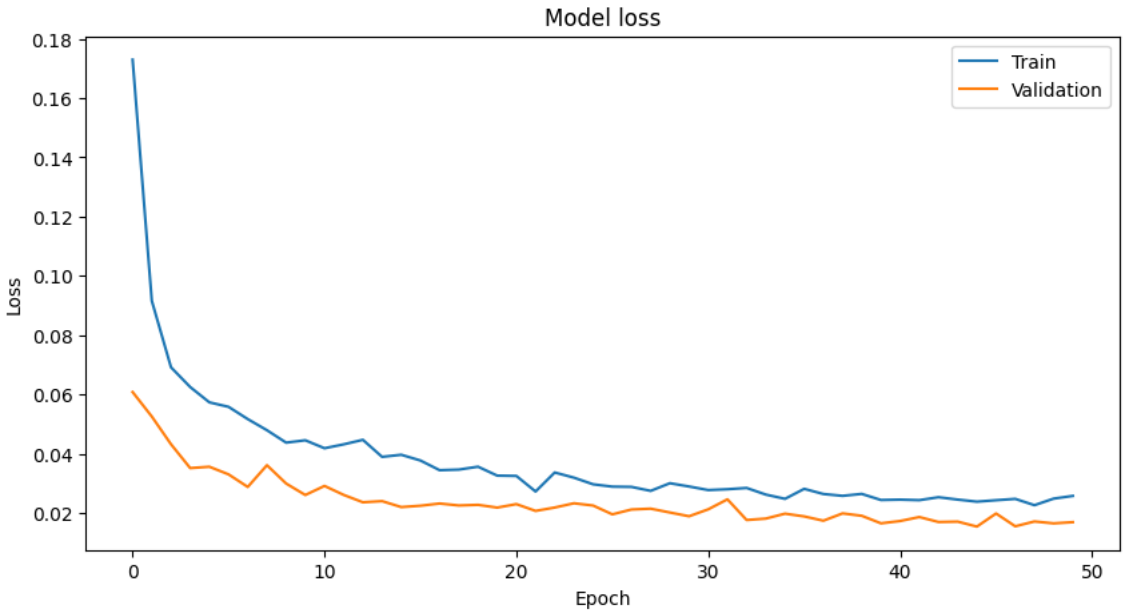

Figure 6. Model Loss - Neural Network

The image represents a plot showing the training and validation loss of a machine learning model over 50 epochs. The training loss starts high but decreases rapidly, indicating the model's learning process. The validation, loss providing an insight into the model's performance on unseen data. At first, both are quickly dropping, showing that the model is learning well and generalizing well. After about 10 epochs the two lines stabilize, with the validation loss always below the line for the training data and thus indicating the model is not overfitting and performs well against new data.

7.6 Sorting and Interpolating Predictions

Smooth curves fitted on the predicted and the current volatility values within the data are generated using interpolation, making it convenient and precise for visualization and other exploratory analyses. Code first concatenates the predictions, specifying flattened=True and

active_columns=True and combines those with the actual values into a DataFrame, indexed by the original test data index. And this DataFrame will be sorted by the index to interpolate the value correctly. Cubic spline interpolation: To interpolate splines (natural cubic spline), it passes a smooth curve through each data point, ensuring continuity and smoothness of up to the second derivative. This approach balances flexibility and smoothness and is applicable in any situation in which we are interpolating between predicted and actual volatility values. The calculated values are then kept in the original data so they can be used subsequently for plotting and analysis.

7.7 Plotting Actual vs Predicted Volatility

This representation helps a bit to understand how the predicted vs actual values of volatility correlate and how is the model performing. Along with different colours and labels, that gives you a plot that looks like this - a line for actual volatility, blue in this case, and a line for predicted volatility, orange. This visual will locates very quickly how good and bad the model was in test period. One easy adjustment to the appearance and readability of a plot is to adjust plot transparency, which can be controlled using the alpha parameter.

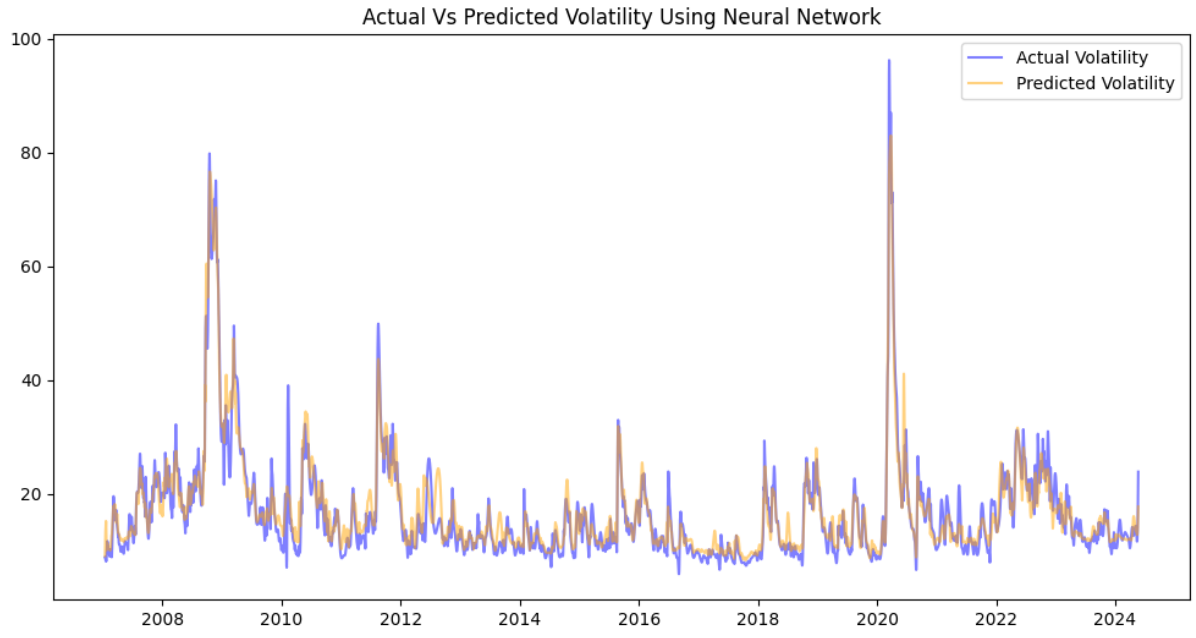

Figure 7. Actual VS Predicted Volatility - Neural Network

The chart is a representation of real versus predicted market volatility over time, taking into account main financial concerns of the last couple of decades that include the 2008 incident and the recent 2020 COVID-19 effect. For those periods, volatility spikes significantly, which shows how instable market environment is. The actual and estimated volatility lines are close,

indicating that the model is able to accurately predict market volatility. Such an incredibly precise prediction is a big plus for financial planning and risk management as it allows investors and analysts to expect about the market actions, decide knowingly and help to avoid risks with higher probability in times of uncertainty. Reliability may be established over a period of time, which then establishes confidence to use the model for future market predictions.

7.8 Analysing Residuals

You also have to look at how the model is doing otherwise you will be literally flying in the dark, one part of which is working with the residuals (the difference between observed and predicted values). Here as well we can identify that what kind of trends or patterns in the errors hidden through residual analysis. Each point in the residual plot does not follow a pattern or show bias. It then subtracts the predicted values from known values and what is left are the residuals, y - pred_y. These residuals are sorted in ascending numerical order. We hope the residuals to be scattered randomly around zero, meaning that no systematic errors exist. Large patterns or biases in the residuals indicate where the model is not so great (and where they can be improved).

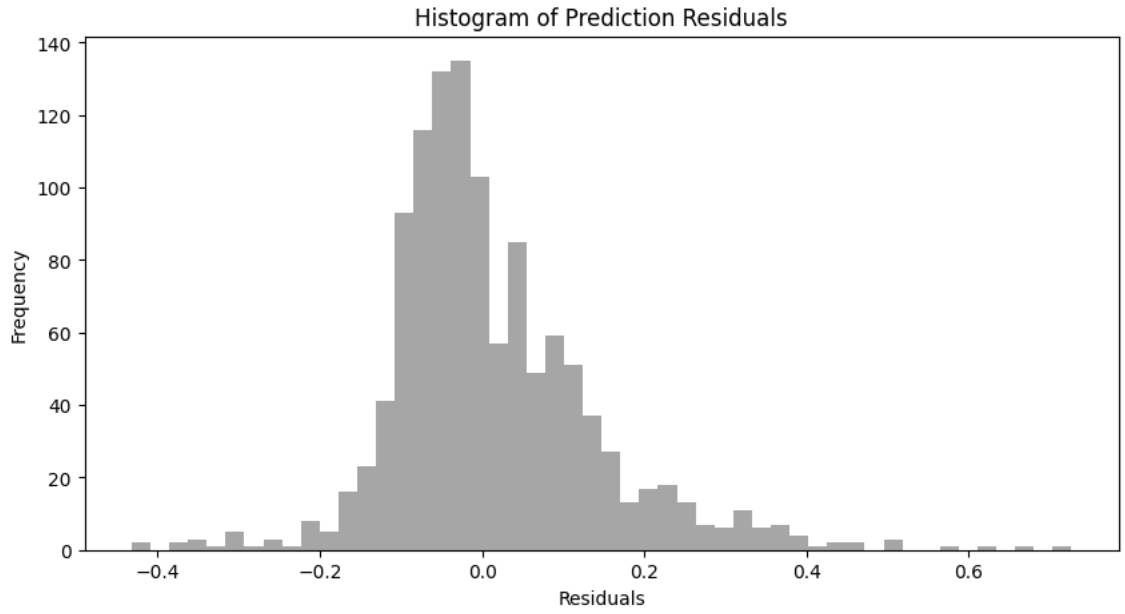

Figure 8. Histogram of Residuals - Neural Network

The histogram shows the distribution of prediction residuals, which are the differences between what you actually got and what researchers expected. Most residuals are close to zero, which says that the model is mostly right in its predictions. Since the residuals spread

out over only a small range, it shows the model is working effectively with very minimal prediction errors. A few residuals are further from zero, showing some larger errors at times. This distribution outcome means that the model can be trusted in almost all future predictions with only very specific exceptions. The latter is critical for the ability to take well-founded decisions and asses risks on the bases of what the model expects.

7.9 Calculating Option Prices

The final section of the code involves calculating the prices of options using the Black-Scholes model. The predicted volatility is scaled and used alongside current prices, strike prices, time to expiration, and risk-free rates to calculate option prices for different strike prices. The Black-Scholes model is a well-known mathematical model used to estimate the price of European-style options. The model takes into account factors such as the current price of the underlying asset, the strike price of the option, the time to expiration, the risk-free interest rate, and the volatility of the underlying asset. The calculated option prices are stored in DataFrames for further analysis and comparison. This part of the code integrates financial modelling with machine learning predictions, demonstrating a practical application of the predictive model in finance.

8. Results

The Random Forest model and Neural Network model were used to predict the option prices for three distinct strike prices (97%, 100%, and 103%) over a period from April 5 to May 22 of 2024. The predictive accuracy of the models was compared against the actual option prices to test their efficacy.

8.1 Random Forest Model Results

8.1.1 Strike 97%

The highest predicted price is 256.94, which happened on April 5. In contrast, the lowest predicted price is anticipated to be 97.88, occurring on May 10. These predictions highlight that the model's forecasts demonstrate significant fluctuation, suggesting that it is highly sensitive to market volatility.

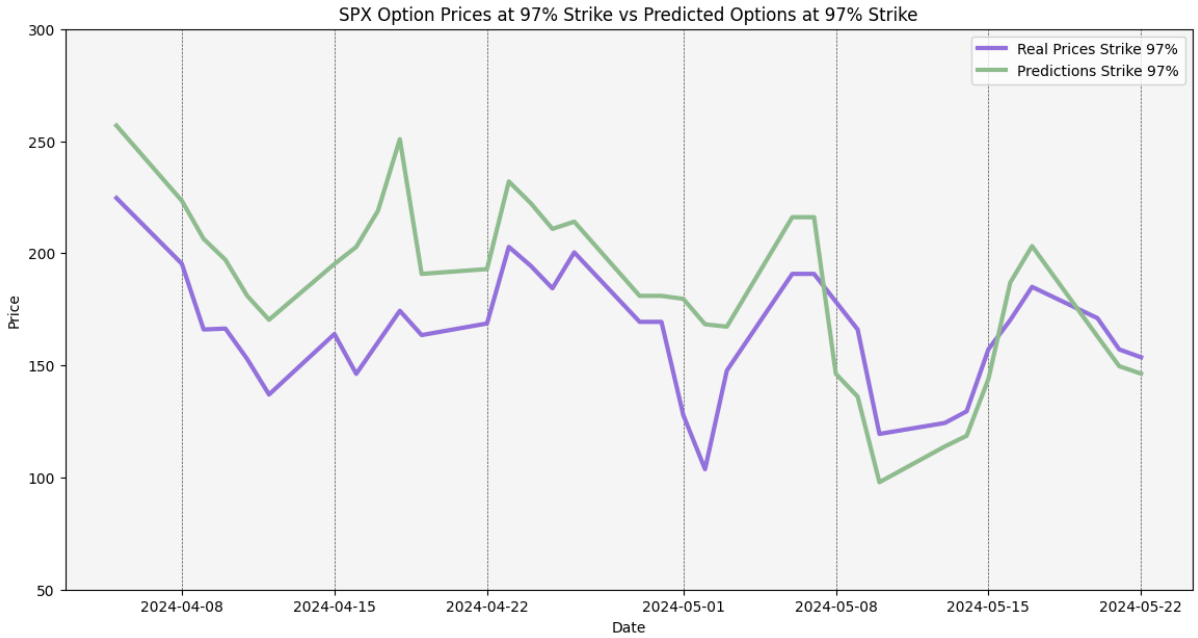

Figure 9. SPX Option Prices vs Predicted Option Prices (Strike 97%) using Random Forest Model

The trend analysis reveals that the graph illustrates the Random Forest model's ability to capture the overall trend of the real prices. However, the model tends to overestimate prices during the initial period and underestimate them towards the end. Additionally, there are significant deviations where the model fails to accurately predict the drop in prices around mid-April, indicating areas where the model could be improved.

8.1.2 Strike 100%

The highest predicted price is 119.47, happened on April 25, while the lowest predicted price is 8.25, occurring on April 18. The observations indicate that these predictions are more stable compared to the 97% strike, showing less extreme volatility while still capturing significant market movements.

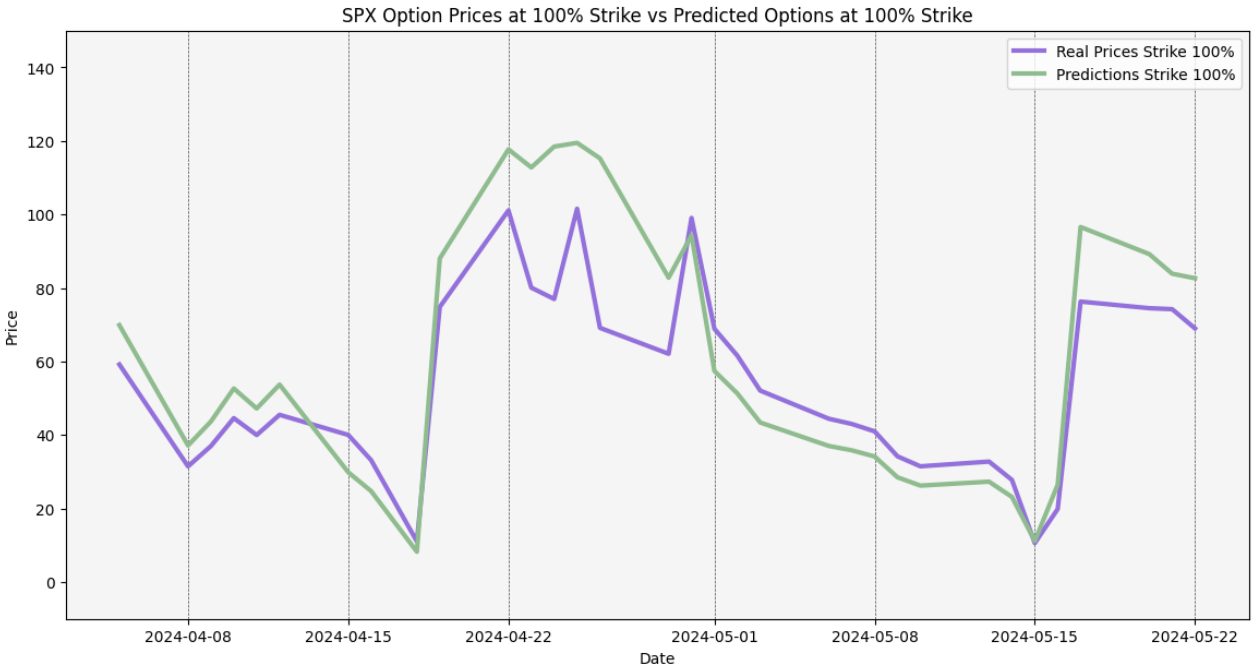

Figure 10. SPX Option Prices vs Predicted Option Prices (Strike 100%) using Random Forest Model

The trend analysis shows that the predictions closely follow the real prices, indicating the model's robustness for this strike price. However, there are minor deviations, particularly during sharp price movements. Significant deviations include slight overestimations observed in late April. Despite these deviations, the model quickly corrects itself, aligning well with real prices by early May.

8.1.3 Strike 103%

The highest predicted price is 38.24, happened on April 23, while the lowest predicted price is 1.14, occurring on April 18. Observations for the 103% strike show that the predictions closely align with actual prices. However, the model tends to underpredict during periods of rapid price increases.

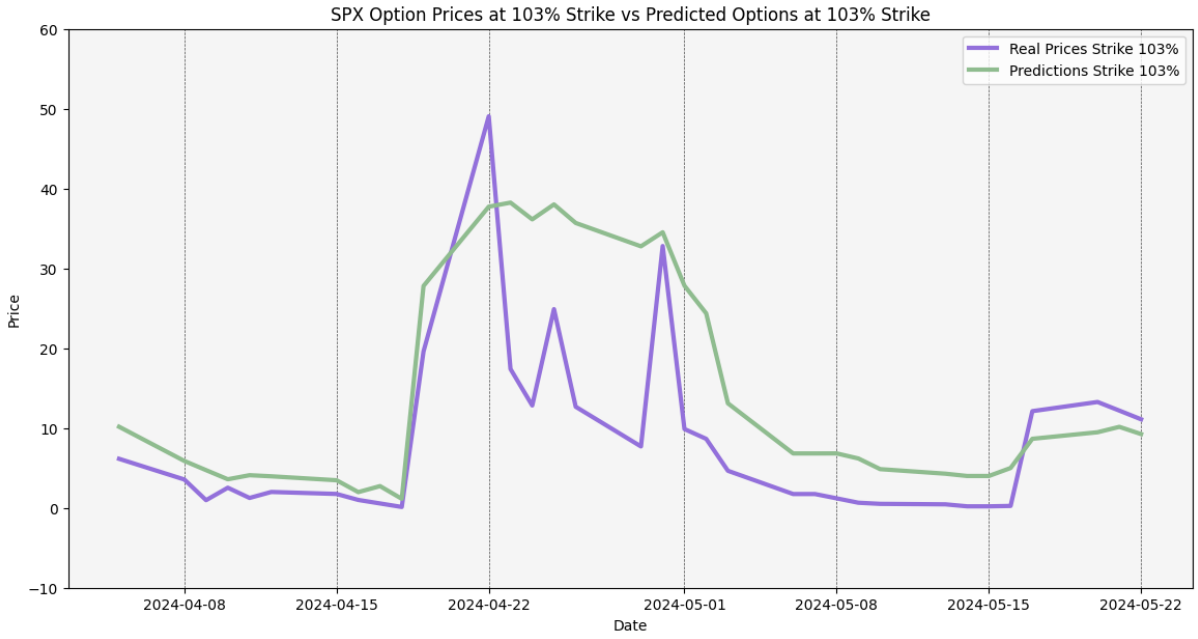

Figure 11. SPX Option Prices vs Predicted Option Prices (Strike 103%) using Random Forest Model

The trend analysis indicates that the predictions are reasonably accurate, maintaining proximity to real prices. However, the model struggles with abrupt price spikes, suggesting a lag in capturing rapid market changes. Significant deviations include notable underpredictions during the price surge in late April, although the overall trend remains consistent.

8.1.4 Error Metric

The Root Mean Square Error (RMSE) was utilized to measure the model's predictive accuracy. The RMSE for the Random Forest model stands at 19.39, reflecting the average magnitude of prediction errors. This metric indicates a moderate level of accuracy, with errors primarily arising during periods of high volatility.

An RMSE of 19.39 points to a moderate prediction error, with the model performing better in stable market conditions. The errors are more pronounced during periods of rapid price changes, suggesting the model's lag in adjusting to new market information.

8.2 Neural Network Model

8.2.1 Strike 97%

The highest predicted price is 232.63, happened on April 5, while the lowest predicted price is 124.92, occurring on May 2, 2024. The observations indicate that the model's predictions display considerable fluctuations, reflecting responsiveness to market changes. However, there are instances where the predictions deviate from actual prices.

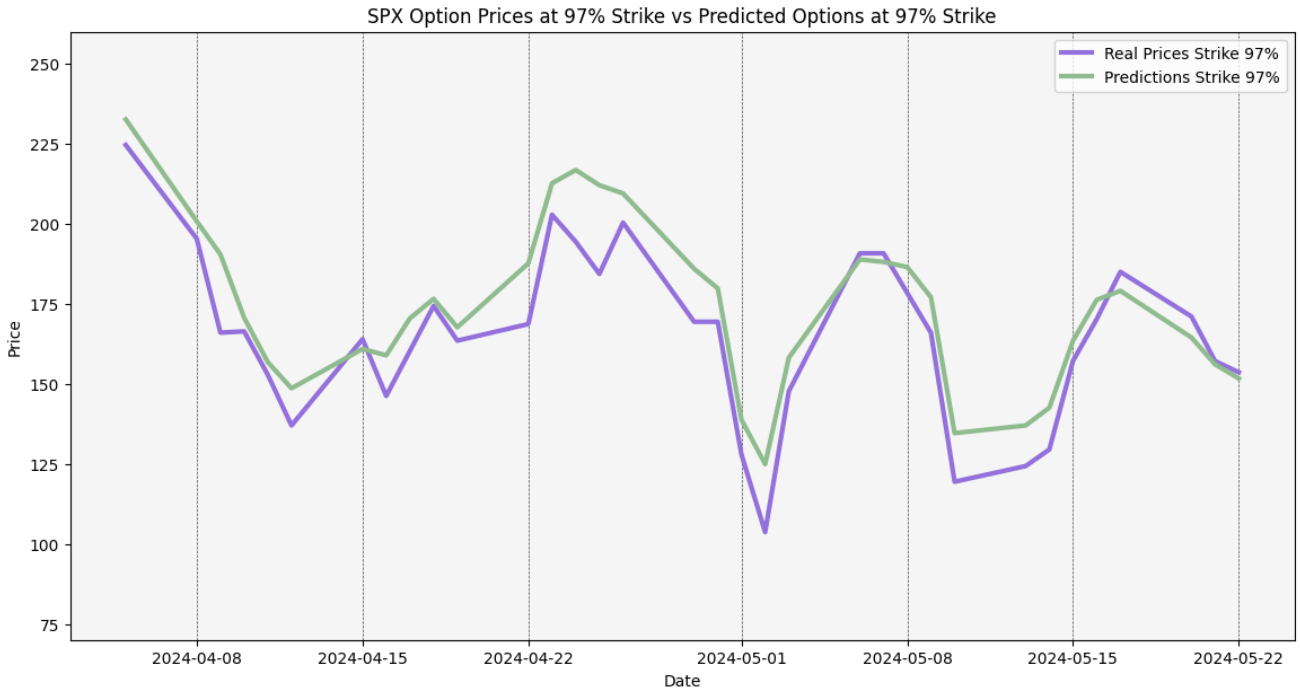

Figure 12. SPX Option Prices vs Predicted Option Prices (Strike 97%) using Neural Network Model

The trend analysis reveals that the graph indicates the Neural Network model generally follows the trend of real prices, though it shows some deviations, especially during periods of rapid price changes. Significant deviations include occasional overpredictions and underpredictions, particularly around mid-April and early May. These deviations highlight areas where the model could be improved.

8.2.2 Strike 100%

The highest predicted price is 113.26, happened on April 25, while the lowest predicted price is 18.86, occurring on April 18. The observations indicate that the predictions show a moderate level of volatility, capturing significant market movements but with some discrepancies.

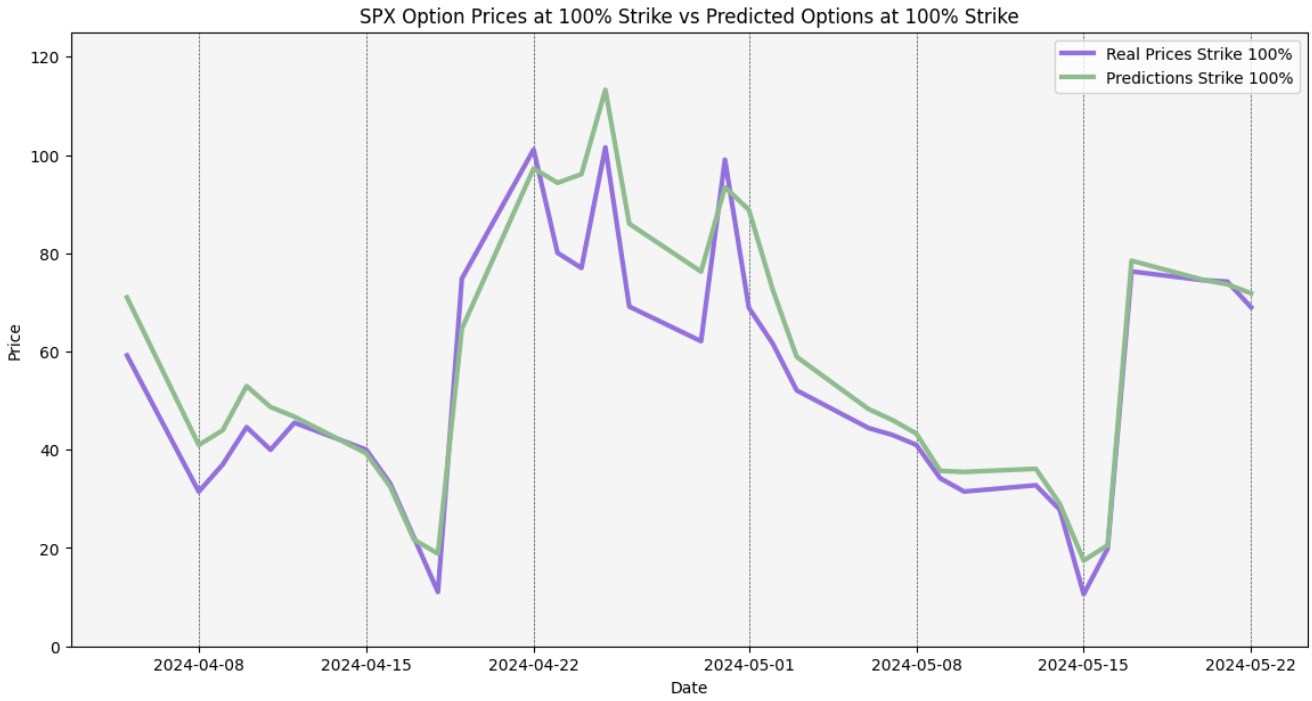

Figure 13. SPX Option Prices vs Predicted Option Prices (Strike 100%) using Neural Network Model

The trend analysis indicates that the model's predictions closely follow real prices, suggesting strong performance with only minor deviations during sharp price movements. Significant deviations include minor overestimations observed in late April; however, the model quickly realigns with actual prices by early May.

8.2.3 Strike 103%

The highest predicted price is 47.23, happened on April 22, while the lowest predicted price is 0.15, occurring on April 18. The observations for the 103% strike show that the predictions closely align with actual prices. However, some underpredictions occur during periods of rapid price spikes.

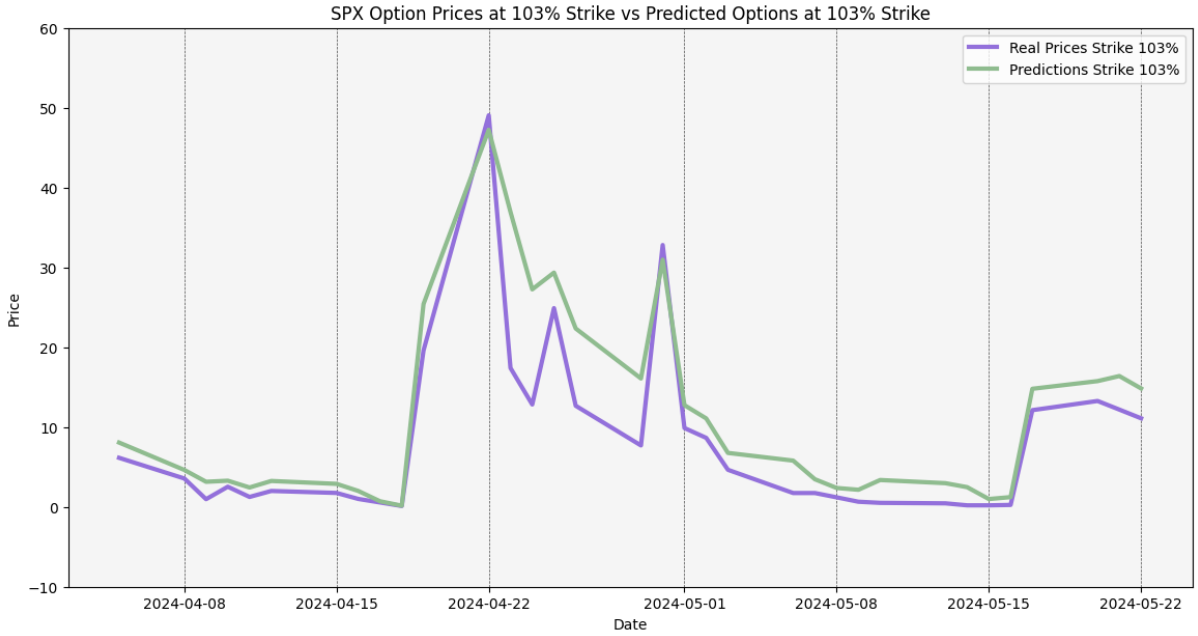

Figure 14. SPX Option Prices vs Predicted Option Prices (Strike 103%) using Neural Network Model

The trend analysis indicates that the predictions are closely aligned with real prices, maintaining overall consistency. However, the model struggles with abrupt price spikes, indicating a lag in capturing rapid market changes. Significant deviations include notable underpredictions during the price surge in late April, although the overall trend remains consistent.

8.2.4 Error Metric

The RMSE for the Neural Network model is 8.63, reflecting a lower average magnitude of prediction errors compared to the Random Forest model.

An RMSE of 8.6357 indicates a lower prediction error compared to the Random Forest model, demonstrating the Neural Network's superior performance. The model performs better in stable market conditions, with more pronounced errors during periods of rapid price changes.

9. Conclusions

Two machine learning models (Random Forest and Neural Network) were used in this study to forecast option prices for three distinct strike prices (97%, 100%, and 103%) over a certain time frame. Comparing the anticipated prices with real market prices allowed us to assess each model's prediction accuracy, trend-following ability, and error metrics.

With an RMSE of 19.39, the Random Forest model showed higher average prediction errors. This shows that the model struggled with precise accuracy, especially during periods of volatile markets, even while it was effective at capturing general market patterns, notably for the 100% and 103% strike prices. For the 97% strike price, the model's performance was less reliable and showed notable differences from actual values.

In contrast, the Neural Network model demonstrated a significantly lower RMSE of 8.63, reflecting its superior performance in minimizing errors. The Neural Network model closely followed the real market prices across all three strike prices, showcasing strong capabilities following trends. Although the model occasionally underpredicted during sharp price spikes, it generally adjusted well to rapid market changes, providing more consistent and reliable predictions compared to the Random Forest model.

The strengths and weaknesses of each model were evident. The Random Forest model effectively captured general market trends and performed well for the 100% and 103% strike prices but had higher prediction errors and struggled during periods of high volatility, particularly for the 97% strike price. On the other hand, the Neural Network model demonstrated superior accuracy with lower RMSE, better trend-following ability, and a more consistent response to rapid price changes, although it still had occasional underpredictions in extreme market conditions.

To enhance predictive performance, a hybrid approach combining the strengths of both models could be explored. Such an approach might leverage the trend-following capability of the Neural Network with the robustness of the Random Forest model, potentially improving overall accuracy.

In conclusion, the Neural Network model demonstrated superior performance in predicting option prices, evidenced by its lower RMSE and better trend-following capabilities. However, the Random Forest model's ability to capture general trends highlights its potential utility. Future work should focus on improving model responsiveness and exploring hybrid approaches to achieve even greater accuracy and reliability in option price prediction.

10.Bibliography

1. Bates, David S. "Jumps and Stochastic Volatility: Exchange Rate Processes Implicit in Deutsche Mark Options." Review of Financial Studies 9, no. 1 (1996): 69-107.
2. Bengio, Yoshua, Ian Goodfellow, and Aaron Courville. Deep Learning. Cambridge, MA: MIT Press, 2016.
3. Bishop, Christopher M. Pattern Recognition and Machine Learning. New York: Springer, 2006.
4. Black, Fischer, and Myron Scholes. "The Pricing of Options and Corporate Liabilities." Journal of Political Economy 81, no. 3 (1973): 637-654.
5. Bloomberg Terminal. "Provides Comprehensive Financial Data, Including Historical Price Data and Economic Indicators." . [https://www.bloomberg.com/professional/solution/bloomberg-terminal/]
6. Chicago Board Options Exchange (CBOE). "For Implied Volatility Data Derived from Options Prices." . [https://www.cboe.com/]
7. CUDA Toolkit Documentation. "For GPU Acceleration Information." . [https://developer.nvidia.com/cuda-toolkit]
8. Derman, Emanuel, and Iraj Kani. "Riding on a Smile." Risk February (1994): 32-39.
9. Dupire, Bruno. "Pricing with a Smile." Risk 7, no. 1 (1994): 18-20.
11. Federal Reserve Economic Data (FRED). "For Macroeconomic Data Like Interest Rates and GDP Growth Rates." . [https://fred.stlouisfed.org/]
12. Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep Learning. Cambridge, MA: MIT Press, 2016.
13. Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. New York: Springer Series in Statistics, 2009.
14. Heston, Steven L. "A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options." Review of Financial Studies 6, no. 2 (1993): 327-343.
15. Hull, John C. Options, Futures, and Other Derivatives. 10th ed. Harlow, UK: Pearson Education Limited, 2017.

16. James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An Introduction to Statistical Learning with Applications in R. New York: Springer, 2013.
17. Joblib Documentation. "For Parallel Processing Techniques." . [https://joblib.readthedocs.io/en/latest/]
18. Lopez de Prado, Marcos. Advances in Financial Machine Learning. Hoboken, NJ: Wiley, 2018.
19. Lopez de Prado, Marcos. Machine Learning for Asset Managers. Cambridge, UK: Cambridge University Press, 2020.
20. Loughran, Tim, and Bill McDonald. "When is a Liability not a Liability? Textual Analysis, Dictionaries, and 10-Ks." The Journal of Finance 66, no. 1 (2011): 35-65.
21. Matplotlib and Seaborn Documentation. "For Data Visualization." . [https://matplotlib.org/] and [https://seaborn.pydata.org/]
22. Merton, Robert C. "Theory of Rational Option Pricing." Bell Journal of Economics and Management Science 4 (1973): 141-183.
23. Murphy, John J. Technical Analysis of the Financial Markets. New York: New York Institute of Finance, 1999.
24. NumPy Documentation. "For Numerical Computations Essential in Financial Modelling." . [https://numpy.org/]
25. Pandas Documentation. "For Data Handling and Manipulation Techniques." . [https://pandas.pydata.org/]
26. Python's asyncio Library. "For Asynchronous Programming to Handle Real-Time Data Feeds." . [https://docs.python.org/3/library/asyncio.html]
27. Scikit-Learn Documentation. "For Machine Learning Algorithms, Data Preprocessing, and Model Validation Techniques." . [https://scikit-learn.org/]
28. TensorFlow Documentation and Keras Documentation. "For Deep Learning Models, Particularly for Implementations Involving Neural Networks." . [https://www.tensorflow.org/] and [https://keras.io/]
29. Tsay, Ruey S. Analysis of Financial Time Series. Hoboken, NJ: Wiley-Interscience, 2005.
30. Yahoo Finance. "A Widely Used Source for Free Historical Stock Price Data." . [https://finance.yahoo.com/]

Annex 1. SPX Real Option Prices

Date	Strike 97%	Strike 100%	Strike 103%
2024 – 04 – 05	224.64	59.20	6.15
2024 – 04 – 08	195.32	31.48	3.58
2024 – 04 – 09	166.00	37.00	0.97
2024 – 04 – 10	166.39	44.60	2.53
2024 – 04 – 11	152.81	40.00	1.25
2024 – 04 – 12	137.00	45.50	2.00
2024 – 04 – 15	163.97	40.03	1.75
2024 – 04 – 16	146.23	33.17	1.00
2024 – 04 – 17	160.29	22.12	0.56
2024 – 04 – 18	174.34	11.06	0.12
2024 – 04 – 19	163.50	74.83	19.60
2024 – 04 – 22	168.68	101.10	49.05
2024 – 04 – 23	202.83	80.08	17.40
2024 – 04 – 24	194.43	77.00	12.83
2024 – 04 – 25	184.35	101.55	24.90
2024 – 04 – 26	200.38	69.15	12.67
2024 – 04 – 29	169.40	62.10	7.70
2024 – 04 – 30	169.40	99.04	32.80
2024 – 05 – 01	127.92	68.90	9.88
2024 – 05 – 02	103.68	61.60	8.65
2024 – 05 – 03	147.88	52.09	4.65
2024 – 05 – 06	190.79	44.43	1.74
2024 – 05 – 07	190.79	43.02	1.74
2024 – 05 – 08	178.40	41.00	1.20
2024 – 05 – 09	166.00	34.20	0.65
2024 – 05 – 10	119.42	31.49	0.51
2024 – 05 – 13	124.31	32.78	0.45
2024 – 05 – 14	129.50	27.80	0.20
2024 – 05 – 15	157.20	10.60	0.20
2024 – 05 – 16	170.20	19.88	0.25
2024 – 05 – 17	185.00	76.30	12.12
2024 – 05 – 20	171.05	74.50	13.27
2024 – 05 – 21	157.10	74.24	12.19
2024 – 05 – 22	153.60	69.00	11.10

Annex 2. Predicted Option Prices using Random Forest Model

Date	Strike 97%	Strike 100%	Strike 103%
2024 – 04 – 05	256.94	69.89	10.16
2024 – 04 – 08	224.40	37.17	5.88
2024 – 04 – 09	206.39	43.68	4.73
2024 – 04 – 10	197.07	52.65	3.59
2024 – 04 – 11	180.99	47.22	4.10
2024 – 04 – 12	170.33	53.71	3.96
2024 – 04 – 15	195.20	29.87	3.46
2024 – 04 – 16	202.82	24.75	1.98
2024 – 04 – 17	218.97	16.50	2.73
2024 – 04 – 18	250.85	8.25	1.14
2024 – 04 – 19	190.78	88.03	27.80
2024 – 04 – 22	192.93	117.69	37.73
2024 – 04 – 23	231.99	112.79	38.24
2024 – 04 – 24	222.38	118.46	36.14
2024 – 04 – 25	210.85	119.47	38.01
2024 – 04 – 26	214.08	115.25	35.69
2024 – 04 – 29	180.98	82.80	32.76
2024 – 04 – 30	180.99	94.32	34.53
2024 – 05 – 01	179.66	57.41	27.83
2024 – 05 – 02	168.31	51.33	24.37
2024 – 05 – 03	167.25	43.41	13.10
2024 – 05 – 06	216.07	37.03	6.82
2024 – 05 – 07	216.07	35.85	6.82
2024 – 05 – 08	146.23	34.16	6.83
2024 – 05 – 09	136.07	28.50	6.19
2024 – 05 – 10	97.89	26.24	4.86
2024 – 05 – 13	113.84	27.31	4.29
2024 – 05 – 14	118.59	23.16	4.00
2024 – 05 – 15	143.96	11.15	4.00
2024 – 05 – 16	186.93	26.51	5.00
2024 – 05 – 17	203.19	96.58	8.66
2024 – 05 – 20	162.90	89.22	9.48
2024 – 05 – 21	149.62	83.89	10.15
2024 – 05 – 22	146.29	82.63	9.25

Annex 3. Predicted Option Prices using Neural Network Model

Date	Strike 97%	Strike 100%	Strike 103%
2024 – 04 – 05	232.63	70.99	8.06
2024 – 04 – 08	200.79	40.97	4.63
2024 – 04 – 09	190.42	43.97	3.15
2024 – 04 – 10	170.65	52.96	3.28
2024 – 04 – 11	156.77	48.70	2.43
2024 – 04 – 12	148.61	46.73	3.25
2024 – 04 – 15	160.90	39.27	2.89
2024 – 04 – 16	158.92	32.51	1.99
2024 – 04 – 17	170.49	21.70	0.71
2024 – 04 – 18	176.56	18.86	0.15
2024 – 04 – 19	167.67	64.64	25.41
2024 – 04 – 22	187.65	97.24	47.23
2024 – 04 – 23	212.67	94.34	36.94
2024 – 04 – 24	216.83	96.08	27.25
2024 – 04 – 25	212.07	113.26	29.35
2024 – 04 – 26	209.50	86.00	22.35
2024 – 04 – 29	185.97	76.26	16.09
2024 – 04 – 30	179.81	93.44	30.97
2024 – 05 – 01	138.79	88.83	12.74
2024 – 05 – 02	124.92	72.55	11.08
2024 – 05 – 03	158.17	58.91	6.76
2024 – 05 – 06	188.87	48.29	5.80
2024 – 05 – 07	188.10	46.06	3.46
2024 – 05 – 08	186.49	43.31	2.37
2024 – 05 – 09	177.07	35.70	2.15
2024 – 05 – 10	134.61	35.49	3.37
2024 – 05 – 13	137.01	36.12	2.97
2024 – 05 – 14	142.54	28.98	2.48
2024 – 05 – 15	163.49	17.45	0.99
2024 – 05 – 16	176.24	20.58	1.23
2024 – 05 – 17	179.07	78.48	14.80
2024 – 05 – 20	164.47	74.58	15.76
2024 – 05 – 21	156.08	73.67	16.39
2024 – 05 – 22	151.68	71.84	14.84

Annex 4. Python Code

```
import numpy as np
import pandas as pd
import yfinance as yf
import talib
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from scipy.stats import norm
from arch import arch_model
from google.colab import drive
from scipy.interpolate import CubicSpline
drive.mount('/content/drive')

#DATA
sp500 = yf.download('^GSPC', start='2007-01-01', end='2024-05-22')
sp500 = sp500[['Close']]
SP500 = sp500.rename(columns={'Close': 'SP500'})

vix = yf.download('^VIX', start='2007-01-01', end='2024-05-22')
vix = vix[['Close']]
vix = vix.rename(columns={'Close': 'VIX'})

vvix = yf.download('^VVIX', start='2007-01-01', end='2024-05-22')
vvix = vvix[['Close']]
vvix = vvix.rename(columns={'Close': 'VVIX'})

gdp = pd.read_csv("/content/drive/MyDrive/gdp_growth.csv",
index_col='DATE', parse_dates=True)
unemployment =
pd.read_csv("/content/drive/MyDrive/unemployment_rate.csv",
index_col='DATE', parse_dates=True)
inflation = pd.read_csv("/content/drive/MyDrive/inflation_rate.csv",
index_col='DATE', parse_dates=True)
tbill = pd.read_csv("/content/drive/MyDrive/t_bill_rate.csv",
index_col='DATE', parse_dates=True)

daily_index = pd.date_range(start='2007-01-01', end='2024-05-22',
freq='D')

SP500 = SP500.reindex(daily_index).interpolate(method='linear',
order=3)
vvix = vvix.reindex(daily_index).interpolate(method='linear', order=3)
gdp = gdp.reindex(daily_index).interpolate(method='linear', order=3)
unemployment =
unemployment.reindex(daily_index).interpolate(method='linear', order=3)
```

```
inflation = inflation.reindex(daily_index).interpolate(method='linear',
order=3)
tbill = (tbill.reindex(daily_index).interpolate(method='linear',
order=3))

data = SP500.join([gdp, unemployment, inflation, tbill, vvix],
how='outer')

data.dropna(inplace=True)

real_option_prices =
pd.read_csv("/content/drive/MyDrive/real_option_prices.csv", sep
=";",index_col='DATE', parse_dates=True)
real_option_prices = real_option_prices.applymap(lambda x:
x.replace(',', '.') if isinstance(x, str) else x)
real_option_prices = real_option_prices.applymap(lambda x:
pd.to_numeric(x, errors='coerce'))

#TECHNICAL INDICATORS
data['MA20'] = talib.MA(data['SP500'], timeperiod=20)
data['MA50'] = talib.MA(data['SP500'], timeperiod=50)
data['MA200'] = talib.MA(data['SP500'], timeperiod=200)
data['RSI'] = talib.RSI(data['SP500'], timeperiod=14)
macd, macdsignal, macdhist = talib.MACD(data['SP500'], fastperiod=12,
slowperiod=26, signalperiod=9)
data['MACD'] = macd
data['MACDSignal'] = macdsignal
data['MACDHist'] = macdhist

plt.figure(figsize=(14,7))
plt.plot(SP500.index, SP500)
plt.title('SP500 Evolution')

plt.figure(figsize=(14,7))
plt.plot(vvix.index, vvix)
plt.title('VVIX Evolution')

plt.figure(figsize=(14,7))
plt.plot(gdp.index, gdp)
plt.title('GDP Growth Rate')

plt.figure(figsize=(14,7))
plt.plot(unemployment.index, unemployment)
plt.title('Unemployment Rate')

plt.figure(figsize=(14,7))
plt.plot(inflation.index, inflation)
plt.title('Inflation Rate')
```

```
plt.figure(figsize=(14,7))
plt.plot(tbill.index, tbill)
plt.title('T-bill Rate')

RANDOM FOREST MODEL
data['log_returns'] = np.log(data['SP500'] / data['SP500'].shift(1))
data['log_returns_scaled'] = data['log_returns'] * 100

#GARCH(1,1) MODEL
model = arch_model(data['log_returns_scaled'].dropna(), vol='Garch',
p=1, q=1)
garch_fit = model.fit(disp="off")
data['garch_vol'] = garch_fit.conditional_volatility
print(data.columns)

#FEATURES
features = ['MA20', 'MA50', 'MA200', 'RSI', 'MACD', 'MACDSignal',
'MACDHist', 'GDP Growth Rate', 'Unemployment Rate', 'Inflation Rate',
'3-Month T-Bill Rate', 'VVIX']
X = data[features].fillna(1e-8)
y = data['garch_vol'].fillna(1e-8)

#RANDOM FOREST
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
model = RandomForestRegressor(n_estimators=1000, random_state=42)
model.fit(X_train, y_train)

# VOLATILIDAD
predicted_volatility = pd.DataFrame(model.predict(X_test),
index=X_test.index, columns=['Predicted Volatility'])
predicted_volatility = predicted_volatility.sort_index()
print(predicted_volatility.shape)
print(predicted_volatility.head())

results = pd.DataFrame({'Actual Volatility': y_test, 'Predicted
Volatility': predicted_volatility['Predicted Volatility']})
print(results.head())

plt.figure(figsize=(14, 7))
plt.plot(data.index, data['garch_vol'], label='GARCH Volatility',
color='blue', alpha=0.5)
plt.scatter(predicted_volatility.index, predicted_volatility['Predicted
Volatility'], label='Predicted Volatility', color='orange', alpha=0.5)
plt.legend()
plt.show()
```

```
#INTERPOLATED RESULTS
cs = CubicSpline(predicted_volatility.index.to_julian_date(),
predicted_volatility['Predicted Volatility'])

data['Interpolated Volatility'] = cs(data.index.to_julian_date())

plt.figure(figsize=(14, 7))
plt.plot(data.index, data['garch_vol'], label='Actual Volatility',
color='blue', alpha=0.5)
plt.plot(data.index, data['Interpolated Volatility'], label='Predicted
Volatility', color='orange', alpha=0.5)
plt.title('Actual Vs Predicted Volatility Using Random Forest')
plt.legend()
plt.show()

Garch_vol = pd.DataFrame(data['garch_vol'])
filtered_Garch_vol =
Garch_vol.loc[Garch_vol.index.intersection(predicted_volatility.index)]

plt.figure(figsize=(14,7))
plt.plot(predicted_volatility*20, label ='Predicted Volatility')
plt.plot(filtered_Garch_vol * 20,label='GARCH Volatility')
plt.plot(vix, label='VIX')
plt.title('Comparison between Volatilities')
plt.legend()

vol = (pd.DataFrame(data['Interpolated Volatility'])
filtered_tbill = tbill.loc[tbill.index.intersection(vol.index)]

def black_scholes(S, K, T, r, sigma):
    d1 = (np.log(S / K) + (r + sigma**2 / 2) * T) / (sigma *
np.sqrt(T))
    pd.DataFrame(d1)
    d2 = d1 - sigma * np.sqrt(T)
    C = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    return C

current_prices = pd.DataFrame(data['SP500'][-len(vol):])
strike_prices_97 = current_prices * 0.97
strike_prices_100 = current_prices * 1
strike_prices_103 = current_prices * 1.03

time_to_expiration = np.array([3/12] * len(vol))
time_to_expiration = pd.DataFrame(time_to_expiration, index=
data.index, columns=['TTM'])

option_prices_97 = pd.DataFrame(black_scholes(current_prices.values,
strike_prices_97.values, time_to_expiration.values,
```

```
filtered_tbill.values, vol.values),index=data.index, columns=['Option
Price Strike 97%'])
option_prices_100 = pd.DataFrame(black_scholes(current_prices.values,
strike_prices_100.values, time_to_expiration.values,
filtered_tbill.values, vol.values),index=data.index, columns=['Option
Price Strike 100%'])
option_prices_103 = pd.DataFrame(black_scholes(current_prices.values,
strike_prices_103.values, time_to_expiration.values,
filtered_tbill.values, vol.values),index=data.index, columns=['Option
Price Strike 103%'])

adjust97 = pd.DataFrame(real_option_prices ['Adj 97% RF'])
adjust100 = pd.DataFrame(real_option_prices ['Adj 100% RF'])
adjust103 = pd.DataFrame(real_option_prices ['Adj 103% RF'])

filtered_tail97 =
pd.DataFrame(option_prices_97.loc[option_prices_97.index.intersection(a
djust97.index)])
results_97 = pd.DataFrame(filtered_tail97.values / adjust97.values,
index= adjust97.index, columns=['Predicted Price Stike 97%'])
filtered_tail100 =
pd.DataFrame(option_prices_100.loc[option_prices_100.index.intersection
(adjust97.index)])
results_100 = pd.DataFrame(filtered_tail100.values / adjust100.values,
index= adjust100.index, columns=['Predicted Price Stike 100%'])
filtered_tail103 =
pd.DataFrame(option_prices_103.loc[option_prices_103.index.intersection
(adjust97.index)])
results_103 = pd.DataFrame(filtered_tail103.values / adjust103.values,
index= adjust103.index, columns=['Predicted Price Stike 103%'])

fig, ax = plt.subplots(figsize=(14, 7))
ax.set_facecolor('whitesmoke')

ax.plot(real_option_prices['Option Strike 97%'], label='Real Prices
Strike 97%',linewidth=3, color = 'mediumpurple')
ax.plot(results_97, label='Predictions Strike 97%', linewidth=3, color
= 'darkseagreen')
ax.set_title('SPX Option Prices at 97% Strike vs Predicted Options at
97% Strike')

ax.set_ylim(50, 300)
ax.set_xlabel('Date')
ax.set_ylabel('Price')
ax.legend()

ax.grid(True, which='both',axis='x', linestyle='--', linewidth=0.5,
color='black', alpha=0.7)
```

```
plt.show()

fig, ax = plt.subplots(figsize=(14, 7))
ax.set_facecolor('whitesmoke')

ax.plot(real_option_prices['Option Strike 100%'], label='Real Prices
Strike 100%',linewidth=3, color = 'mediumpurple')
ax.plot(results_100, label='Predictions Strike 100%', linewidth=3,
color = 'darkseagreen')
ax.set_title('SPX Option Prices at 100% Strike vs Predicted Options at
100% Strike')

ax.set_ylim(-10, 150)
ax.set_xlabel('Date')
ax.set_ylabel('Price')
ax.legend()

ax.grid(True, which='both',axis='x', linestyle='--', linewidth=0.5,
color='black', alpha=0.7)


plt.show()

fig, ax = plt.subplots(figsize=(14, 7))
ax.set_facecolor('whitesmoke')

ax.plot(real_option_prices['Option Strike 103%'], label='Real Prices
Strike 103%',linewidth=3, color = 'mediumpurple')
ax.plot(results_103, label='Predictions Strike 103%', linewidth=3,
color = 'darkseagreen')
ax.set_title('SPX Option Prices at 103% Strike vs Predicted Options at
103% Strike')

ax.set_ylim(-10, 60)
ax.set_xlabel('Date')
ax.set_ylabel('Price')
ax.legend()

ax.grid(True, which='both',axis='x', linestyle='--', linewidth=0.5,
color='black', alpha=0.7)


plt.show()

real_strike97 = pd.DataFrame(real_option_prices['Option Strike 97%'])
real_strike100 = pd.DataFrame(real_option_prices['Option Strike 100%'])
real_strike103 = pd.DataFrame(real_option_prices['Option Strike 103%'])
```

```
#Mean Square Error
mse_97 = (pd.DataFrame(real_strike97.values - results_97.values,
index=real_strike97.index, columns=['MSE strike 97%']) ** 2).mean()
print("Error Cuadrático Medio (MSE):")
print(mse_97)

mse_100 = (pd.DataFrame(real_strike100.values - results_100.values,
index=real_strike100.index, columns=['MSE strike 100%']) ** 2).mean()
print("Error Cuadrático Medio (MSE):")
print(mse_100)

mse_103 = (pd.DataFrame(real_strike103.values - results_103.values,
index=real_strike103.index, columns=['MSE strike 103%']) ** 2).mean()
print("Error Cuadrático Medio (MSE):")
print(mse_103)
#Root Mean Square Error
rmse_97 = np.sqrt(mse_97)
print("Raíz del Error Cuadrático Medio (RMSE):")
print(rmse_97)

rmse_100 = np.sqrt(mse_100)
print("Raíz del Error Cuadrático Medio (RMSE):")
print(rmse_100)

rmse_103 = np.sqrt(mse_103)
print("Raíz del Error Cuadrático Medio (RMSE):")
print(rmse_103)

print("")
mean = pd.DataFrame(np.mean([rmse_97.values, rmse_100.values,
rmse_103.values], axis=0),columns=['RMSE'])
print("RMSE Promedio para el modelo Random Forest:")
print(mean)
```

NEURAL NETWORK MODEL

```
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.metrics import mean_squared_error, r2_score

# NORMALIZE
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

data_NN = data.iloc[10:]
```

```
# NEURAL MODEL
model = Sequential()
model.add(Dense(128, input_dim=X_train_scaled.shape[1],
activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='linear'))

#COMPILE MODEL
model.compile(loss='mean_squared_error', optimizer='adam')
history = model.fit(X_train_scaled, y_train, validation_split=0.2,
epochs=50, batch_size=32, verbose=1)

# PREDICTING
predictions = model.predict(X_test_scaled)
mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
print(f'Mean Squared Error: {mse}')
print(f'R^2 Score: {r2}')

# PLOT LOSS VALUES
plt.figure(figsize=(10, 5))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()

predictions = model.predict(X_test_scaled).flatten()
results_df = pd.DataFrame({'Actual Volatility': y_test, 'Predicted
Volatility': predictions}, index=y_test.index)
df_sorted = results_df.sort_index()
plt.plot(df_sorted, color = 'red')
plt.plot(vix, color = 'red', label = 'vix')

#INTERPOLATION
cs1 = CubicSpline(df_sorted.index, df_sorted['Predicted Volatility'])
cs2 = CubicSpline(df_sorted.index, df_sorted['Actual Volatility'])

data['Interpolated Predicted Volatility NN'] = cs1(data.index)
data['Interpolated Actual Volatility NN'] = cs2(data.index)

Actual_Volatility_NN = pd.DataFrame(data['Interpolated Actual
Volatility NN']).iloc[10:]
```

```
Predicted_Volatility_NN = pd.DataFrame(data['Interpolated Predicted
Volatility NN']).iloc[10:]

plt.figure(figsize=(12, 6))
plt.plot(data_NN.index, Actual_Volatility_NN, label='Actual
Volatility', color='blue', alpha=0.5)
plt.plot(data_NN.index, Predicted_Volatility_NN, label='Predicted
Volatility', color='orange', alpha=0.5)
plt.legend()
plt.show()

results_df['Residuals'] = results_df['Actual Volatility'] -
results_df['Predicted Volatility']

plt.figure(figsize=(10, 5))
plt.hist(results_df['Residuals'], bins=50, color='gray', alpha=0.7)
plt.title('Histogram of Prediction Residuals')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.show()

vol1 = pd.DataFrame(Predicted_Volatility_NN/3)

current_prices_NN = pd.DataFrame(data['SP500'][-len(vol1):])
strike_prices_97_NN = current_prices_NN * 0.97
strike_prices_100_NN = current_prices_NN * 1
strike_prices_103_NN = current_prices_NN * 1.03

time_to_expiration_NN = np.array([3/12] * len(vol1))
time_to_expiration_NN = pd.DataFrame(time_to_expiration, index=
data_NN.index, columns=['TTM'])

filtered_tbill_NN = filtered_tbill.iloc[10:]

option_prices_97NN =
pd.DataFrame(black_scholes(current_prices_NN.values,
strike_prices_97_NN.values, time_to_expiration_NN.values,
filtered_tbill_NN.values,  vol1.values),index=data_NN.index,
columns=['Option Price Strike 97%'])
option_prices_100NN =
pd.DataFrame(black_scholes(current_prices_NN.values,
strike_prices_100_NN.values, time_to_expiration_NN.values,
filtered_tbill_NN.values,  vol1.values),index=data_NN.index,
columns=['Option Price Strike 100%'])
option_prices_103NN =
pd.DataFrame(black_scholes(current_prices_NN.values,
strike_prices_103_NN.values, time_to_expiration_NN.values,
```

```
filtered_tbill_NN.values,  vol1.values),index=data_NN.index,
columns=['Option Price Strike 103%'])

real_strike97 = pd.DataFrame(real_option_prices['Option Strike 97%'])
real_strike100 = pd.DataFrame(real_option_prices['Option Strike 100%'])
real_strike103 = pd.DataFrame(real_option_prices['Option Strike 103%'])

adjust97NN = pd.DataFrame(real_option_prices ['Adj 97% NN'])
adjust100NN = pd.DataFrame(real_option_prices ['Adj 100% NN'])
adjust103NN = pd.DataFrame(real_option_prices ['Adj 103% NN'])

filtered_tail97NN =
pd.DataFrame(option_prices_97NN.loc[option_prices_97NN.index.intersecti
on(real_strike97.index)])
results_97NN = pd.DataFrame(filtered_tail97NN.values /
adjust97NN.values, index= adjust97NN.index, columns=['Predicted Price
Stike 97%'])
filtered_tail100NN =
pd.DataFrame(option_prices_100NN.loc[option_prices_100NN.index.intersec
tion(real_strike100.index)])
results_100NN = pd.DataFrame(filtered_tail100NN.values /
adjust100NN.values, index= adjust100NN.index, columns=['Predicted Price
Stike 100%'])
filtered_tail103NN =
pd.DataFrame(option_prices_103NN.loc[option_prices_103NN.index.intersec
tion(real_strike103.index)])
results_103NN = pd.DataFrame(filtered_tail103NN.values /
adjust103NN.values, index= adjust103NN.index, columns=['Predicted Price
Stike 103%'])

fig, ax = plt.subplots(figsize=(14, 7))
ax.set_facecolor('whitesmoke')

ax.plot(real_strike97, label='Real Prices Strike 97%',linewidth=3,
color = 'mediumpurple')
ax.plot(results_97NN, label='Predictions Strike 97%', linewidth=3,
color = 'darkseagreen')
ax.set_title('SPX Option Prices at 97% Strike vs Predicted Options at
97% Strike')

ax.set_ylim(70, 260)
ax.set_xlabel('Date')
ax.set_ylabel('Price')
ax.legend()

ax.grid(True, which='both',axis='x', linestyle='--', linewidth=0.5,
color='black', alpha=0.7)
```

```
plt.show()

fig, ax = plt.subplots(figsize=(14, 7))
ax.set_facecolor('whitesmoke')

ax.plot(real_strike100, label='Real Prices Strike 100%',linewidth=3,
color = 'mediumpurple')
ax.plot(results_100NN, label='Predictions Strike 100%', linewidth=3,
color = 'darkseagreen')
ax.set_title('SPX Option Prices at 100% Strike vs Predicted Options at
100% Strike')

ax.set_ylim(0, 125)
ax.set_xlabel('Date')
ax.set_ylabel('Price')
ax.legend()

ax.grid(True, which='both',axis='x', linestyle='--', linewidth=0.5,
color='black', alpha=0.7)


plt.show()

fig, ax = plt.subplots(figsize=(14, 7))
ax.set_facecolor('whitesmoke')

ax.plot(real_strike103, label='Real Prices Strike 103%',linewidth=3,
color = 'mediumpurple')
ax.plot(results_103NN, label='Predictions Strike 103%', linewidth=3,
color = 'darkseagreen')
ax.set_title('SPX Option Prices at 103% Strike vs Predicted Options at
103% Strike')

ax.set_ylim(-10, 60)
ax.set_xlabel('Date')
ax.set_ylabel('Price')
ax.legend()

ax.grid(True, which='both',axis='x', linestyle='--', linewidth=0.5,
color='black', alpha=0.7)


plt.show()

#Mean Square Error
mse_97NN = (pd.DataFrame(real_strike97.values - results_97NN.values,
index=real_strike97.index, columns=['MSE strike 97%']) ** 2).mean()
print("Error Cuadrático Medio (MSE):")
print(mse_97NN)
```

```
mse_100NN = (pd.DataFrame(real_strike100.values - results_100NN.values,
index=real_strike100.index, columns=['MSE strike 100%']) ** 2).mean()
print("Error Cuadrático Medio (MSE):")
print(mse_100NN)

mse_103NN = (pd.DataFrame(real_strike103.values - results_103NN.values,
index=real_strike103.index, columns=['MSE strike 103%']) ** 2).mean()
print("Error Cuadrático Medio (MSE):")
print(mse_103NN)

#Root Mean Square Error
rmse_97NN = np.sqrt(mse_97NN)
print("Raíz del Error Cuadrático Medio (RMSE):")
print(rmse_97NN)

rmse_100NN = np.sqrt(mse_100NN)
print("Raíz del Error Cuadrático Medio (RMSE):")
print(rmse_100NN)

rmse_103NN = np.sqrt(mse_103NN)
print("Raíz del Error Cuadrático Medio (RMSE):")
print(rmse_103NN)

print("")
mean = pd.DataFrame(np.mean([rmse_97NN.values, rmse_100NN.values,
rmse_103NN.values], axis=0),columns=['RMSE'])
print("RMSE Promedio para el modelo Random Forest")
print(mean)
```

Appendix 1. AI Declaration

Declaration of Use of Generative Artificial Intelligence Tools in Advantere School of Management's Capstone Projects.

At Advantere School of Management, we consider that ChatGPT or other similar tools are very useful in academic life, although their use is always under the student's responsibility as the responses provided may not be accurate.

By this statement, I (Emanuele Ospina), a student of Advantere School of Management's Master in Finance, upon presenting my Capstone Project titled "Enhancing Option Pricing Using Machine Learning and Black-Scholes Model," declare that I have used the Generative Artificial Intelligence tool ChatGPT or other similar AI tools only in the context of the activities described below:

1. Multidisciplinary Studies:To understand perspectives from other communities on topics of a multidisciplinary nature.
2. Literary and Language Style Corrector:To improve the linguistic and stylistic quality of the text.
3. Reviewer: To receive suggestions on how to improve and perfect the work with different levels of rigor.

I affirm that all the information and content presented in this work are the product of my individual research and effort except where otherwise indicated and appropriate credits have been given (I have included the appropriate references in the Capstone Project Thesis document and have explicitly stated where ChatGPT or other similar tools have been used). I am aware of the academic and ethical implications of presenting non-original work and accept the consequences of any violation of this declaration.

Date: [05/07/2024]

Signature: _____

Appendix 2. Authorization for Digital Publication of Capstone Project

AUTHORIZATION FOR DIGITALIZATION, STORAGE AND DISSEMINATION IN THE NETWORK OF CAPSTONE PROJECTS

1. Declaration of authorship and accreditation thereof.

The author Mr. /Ms._____Emanuele Ospina Castellanos_____

HEREBY DECLARES that he/she owns the intellectual property rights regarding the piece of work: ____" Enhancing Option Pricing Using Machine Learning and Black-Scholes Model " _____that this is an original piece of work, and that he/she holds the status of author, in the sense granted by the Intellectual Property Law.

2. Subject matter and purpose of this assignment.

With the aim of disseminating the aforementioned piece of work as widely as possible using the Advantere School of Management Institutional Repository the author hereby **GRANTS** Advantere School of Management, on a royalty-free and non-exclusive basis, for the maximum legal term and with universal scope, the digitization, archiving, reproduction, distribution and public communication rights, including the right to make it electronically available, as described in the Intellectual Property Law. Transformation rights are assigned solely for the purposes described in a) of the following section.

3. Transfer and access terms

Without prejudice to the ownership of the work, which remains with its author, the transfer of rights covered by this license enables:

a) Transform it in order to adapt it to any technology suitable for sharing it online, as well as including metadata to register the piece of work and include "watermarks" or any other security or protection system.

b) Reproduce it in any digital medium in order to be included on an electronic database, including the right to reproduce and store the work on servers for the purposes of guaranteeing its security, maintaining it and preserving its format.

c) Communicate it, by default, by means of an institutional open archive, which has open and cost-free online access.

d) Any other way of access (restricted, embargoed, closed) shall be explicitly requested and requires that good cause be demonstrated.

e) Assign these pieces of work a Creative Commons license by default.

f) Assign these pieces of work a HANDLE (*persistent* URL). by default.

4. Copyright.

The author, as the owner of a piece of work, has the right to:

a) Have his/her name clearly identified by the Advantere School of Management as the author

b) Communicate and publish the work in the version assigned and in other subsequent versions using any medium.

c) Request that the work be withdrawn from the repository for just cause.

d) Receive reliable communication of any claims third parties may make in relation to the work and, in particular, any claims relating to its intellectual property rights.

5. Duties of the author.

The author agrees to:

a) Guarantee that the commitment undertaken by means of this official document does not infringe any third party rights, regardless of whether they relate to industrial or intellectual property or any other type.

b) Guarantee that the content of the work does not infringe any third party honor, privacy or image rights.

c) Take responsibility for all claims and liability, including compensation for any damages, which may be brought against the Advantere School of Management by third parties who believe that their rights and interests have been infringed by the assignment.
d) Take responsibility in the event that the institutions are found guilty of a rights infringement regarding the work subject to assignment.

6. Institutional Repository purposes and functioning.

The work shall be made available to the users so that they may use it in a fair and respectful way with regards to the copyright, according to the allowances given in the relevant legislation, and for study or research purposes, or any other legal use. With this aim in mind, the Advantere School of Management undertakes the following duties and reserves the following powers:

a) Advantere School of Management shall inform the archive users of the permitted uses; however, it shall not guarantee or take any responsibility for any other subsequent ways the work may be used by users, which are non-compliant with the legislation in force. Any subsequent use, beyond private copying, shall require the source to be cited and authorship to be recognized, as well as the guarantee not to use it to gain commercial profit or carry out any derivative works.
b) Advantere School of Management shall not review the content of the works, which shall at all times fall under the exclusive responsibility of the author and it shall not be obligated to take part in lawsuits on behalf of the author in the event of any infringement of intellectual property rights deriving from storing and archiving the works. The author hereby waives any claim against the Advantere School of Management due to any way the users may use the works that is not in keeping with the legislation in force.
c) The Advantere School of Management shall adopt the necessary measures to safeguard the work in the future.
d) The Advantere School of Management reserves the right to withdraw the work, after notifying the author, in sufficiently justified cases, or in the event of third party claims.

Madrid, on …5.. of ……July……..., …2024….

HEREBY ACCEPTS

Signed……

Reasons for requesting the restricted, closed or embargoed access to the work in the Institution's Repository